(12)特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局

(43) 国際公開日
2004 年4 月22 日 (22.04.2004)

PCT

(10) 国際公開番号
WO 2004/034698 A1

(51) 国際特許分類[7]: H04N 5/44, 5/93

(21) 国際出願番号: PCT/JP2003/012932

(22) 国際出願日: 2003 年10 月9 日 (09.10.2003)

(25) 国際出願の言語: 日本語

(26) 国際公開の言語: 日本語

(30) 優先権データ:
特願2002-296234 2002 年10 月9 日 (09.10.2002) JP

(71) 出願人 (米国を除く全ての指定国について): 松下電器産業株式会社 (MATSUSHITA ELECTRIC INDUSTRIAL CO., LTD.) [JP/JP]; 〒571-8501 大阪府 門真市 大字門真１００６ Osaka (JP).

(72) 発明者; および
(75) 発明者/出願人 (米国についてのみ): 堀井 幸 (HORII,Yuki) [JP/JP]; 〒567-0803 大阪府 茨木市 中総持寺町５−４−３０５ Osaka (JP). 川上 義雄 (KAWAKAMI,Yoshio) [JP/JP]; 〒571-0079 大阪府 門真市 野里町６−１６−２０１ Osaka (JP). 脇 康 (WAKI,Yasushi) [JP/JP]; 〒619-0232 京都府 相楽郡 精華町桜ヶ丘１−３０−１６ Kyoto (JP).

(74) 代理人: 小笠原 史朗 (OGASAWARA,Shiro); 〒564-0053 大阪府 吹田市 江の木町３番１１号 第３ロンチェビル Osaka (JP).

(81) 指定国 (国内): CN, JP, US.

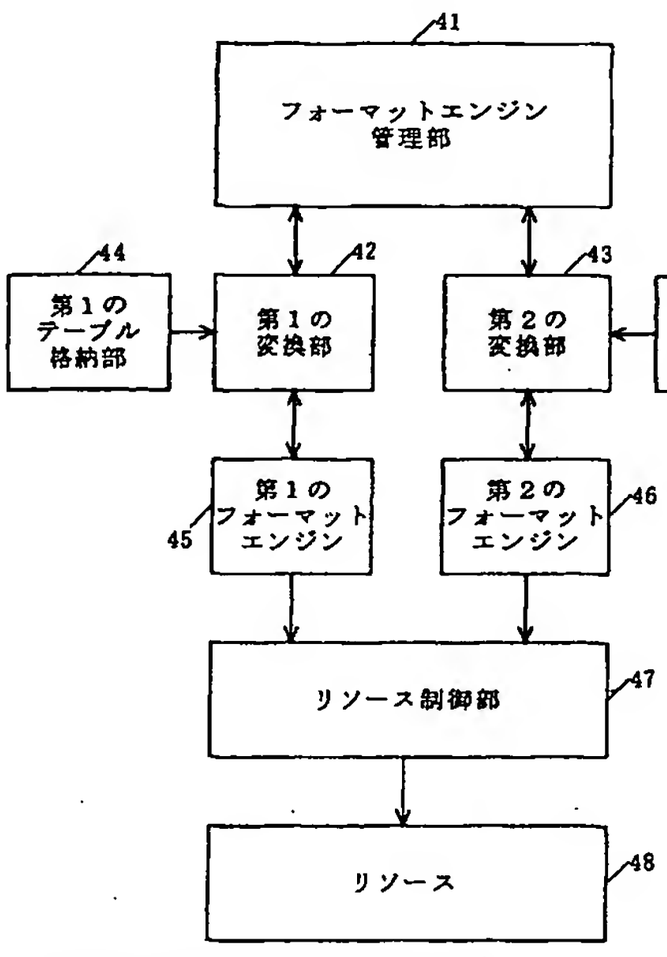
(84) 指定国 (広域): ヨーロッパ特許 (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR).

添付公開書類:
— 国際調査報告書

[続葉有]

(54) Title: INFORMATION PROCESSOR

(54) 発明の名称: 情報処理装置



41...FORMAT ENGINE MANAGING SECTION
42...FIRST CONVERTING SECTION
43...SECOND CONVERTING SECTION
44...FIRST TABLE STORING SECTION
45...FIRST FORMAT ENGINE
46...SECOND FORMAT ENGINE
47...RESOURCE CONTROL SECTION
48...RESOURCE
49...SECOND TABLE STORING SECTION

(57) Abstract: An information processor in which addition or alteration of a format engine can be dealt with readily. The information processor comprises a format engine managing means, and an operation control means. The format engine managing means previously defines a common state where the operating state of the format engine is defined by a representation common to all format engines, and manages the operation of each format engine. The operation control means previously defines the correspondence between the common state and the individual state where the operating state of the format engine is defined by a representation different for each format engine and controls the operation of the format engine such that an arbitrary individual state is brought about.

(57) 要約: フォーマットエンジンの追加や変更に容易に対応することができる情報処理装置を提供する。情報処理装置は、フォーマットエンジン管理手段と、動作制御手段とを備えている。フォーマットエンジン管理手段は、フォーマットエンジンの動作状態を全てのフォーマットエンジンに共通の表現によって規定した共通状態を予め定義しておき、各フォーマットエンジンの動作を管理する。動作制御手段は、各フォーマットエンジンに対応して設けられ、共通状態と、フォーマットエンジンの動作状態を各フォーマットエンジン毎に異なる表現によって規定した動作状態である個別状態との対応を予め定義しておき、任意の個別状態となるようにフォーマットエンジンの動作を制御する。

WO 2004/034698 A1

２文字コード及び他の略語については、定期発行される各*PCT*ガゼットの巻頭に掲載されている「コードと略語のガイダンスノート」を参照。

明細書

情報処理装置

技術分野

本発明は、情報処理装置に関し、より特定的には、ＪａｖａやＨＴＭＬ等の特定のフォーマットで記述されたアプリケーションを実行するフォーマットエンジンの動作を管理する情報処理装置に関する。

背景技術

近年、デジタルテレビ、携帯情報端末、携帯電話といった情報処理装置において、ＪａｖａプログラムやＨＴＭＬコンテンツ等の特定のフォーマットで記述されたアプリケーションを実行することが盛んに行われている。ここで、このようなアプリケーションの実行やデータの再生を行うためのソフトウェアをフォーマットエンジンと呼ぶ。フォーマットエンジンとは、例えば、上記Ｊａｖａプログラムを実行するためのＪａｖａミドルウェアやＨＴＭＬコンテンツを表示するためのブラウザである。これらのフォーマットエンジンの他にも、現在パーソナルコンピュータ上で実行されているＰＤＦデータや音楽データ等を再生するためのフォーマットエンジンも、上記デジタルテレビ等において実行されるようになるであろう。

一般的に、上記デジタルテレビ等の情報処理装置に格納されているフォーマットエンジンは１つである。図９４は

、従来のデジタル放送受信機の構成図の一例である。このデジタル放送受信機は、放送データに含まれるアプリケーション（Ｊａｖａプログラム）を実行し、管理するものである。図９４において、デジタル放送受信機９４０２は、テレビモニタ９４０１に接続されている。また、デジタル放送受信機９４０２は、ディスプレイマネージャ９４１１と、アプリケーションマネージャ９４１２と、ルール９４１４と、信号モニタ９４１３とを備えている。

図９４において、デジタル放送受信機９４０２の信号モニタ９４１３は、放送データストリームを受信する。信号モニタ９４１３は、放送データストリームに含まれるアプリケーションが含まれるか否かを判断し、アプリケーションの有無を示す信号データをアプリーケーションマネージャ９４１２へ出力する。アプリケーションマネージャ９４１２は、入力される信号データに基づいて、放送データストリームにアプリケーションが含まれるか否かを判断する。そして、放送データストリームにアプリケーションが含まれる場合、アプリケーションマネージャ９４１２は、当該アプリケーションをアプリケーションを読み込み実行する。アプリケーションが実行されると、テレビモニタに表示されるべきディスプレイ情報がディスプレイコンテキストとしてディスプレイマネージャ９４１１へ提供される。ディスプレイマネージャ９４１１は、適切なディスプレイ情報をテレビモニタ９４０１に提供する。ここで、アプリケーションマネージャ９４１２は、ルール９４１４に従って動作し、各アプリケーションの状態をロード状態、一時

停止状態、アクティブ状態、および抹消状態のいずれかに遷移させる。ルールは、例えば、「１度に１つのアプリケーションのみをアクティブ状態にする」や、「１度に１つのアプリケーションのみを表示できる」等である。デジタル放送受信機９４０２は、このようなルールを用いることによって複数のアプリケーションを管理することができる。

一方、デジタルテレビ、携帯情報端末、携帯電話といった情報処理装置が複数のフォーマットエンジンを備える場合、それら複数のフォーマットエンジンを管理する技術も考えられている。このようなデジタルテレビ、携帯情報端末、携帯電話といった情報処理装置では、ＣＰＵに比べるとフォーマットエンジンを実行する際に用いられるリソースが限られているという側面がある。リソースとは、例えばメモリのように、フォーマットエンジンによって利用されるハードウェアをいう。例えば、デジタルテレビが有するリソースには、メモリの他、放送波を受信しデータを復号するためのチューナ、放送波に含まれる映像や音声をデコードするためのデコーダ、インターネットに接続するためのネットワークインターフェース等がある。フォーマットエンジンを実行する際には、実行されるフォーマットエンジンが利用するリソースを確保する必要があるので、上記のようなリソースが限られた情報処理装置においては、各フォーマットエンジンの実行を管理することが行われている。例えば、デジタルテレビでは、各フォーマットエンジンを切り替えて実行する処理を行うものがある。以下、

従来のデジタルテレビにおいて複数のフォーマットエンジンを管理する技術を説明する。

図９５は、従来のデジタルテレビにおいて用いられるミドルウェア切替装置の構成図の一例である。このミドルウェア切替装置は、複数のミドルウェア（フォーマットエンジン）を切り替えて実行する。図９５において、ミドルウェア切替装置は、ミドルウェア決定部９５０１と、ミドルウェア読込部９５０２と、ミドルウェア実行部９５０３と、ミドルウェア格納部９５０４とを備えている。

図９５において、ミドルウェア格納部９５０４は、第１～第ｎのミドルウェアを格納している。ミドルウェア決定部９５０１は、第１～第ｎのミドルウェアの内、実行すべきミドルウェアを決定する。ミドルウェア読込部９５０２は、ミドルウェア決定部９５０１によって決定されたミドルウェアをミドルウェア格納部９５０４から読み込む。ミドルウェア実行部９５０３は、ミドルウェア読込部９５０２が読み込んだミドルウェアを実行する。以上の構成によって、第１～第ｎのミドルウェアは切り替えて実行される。これによって、図９５に示す装置は、複数のミドルウェアを１つのデジタルテレビ上に共存させることを可能としている。

発明の開示

しかし、図９５に示した構成では、ミドルウェアの追加や変更への対応が困難であった。以下、詳細を説明する。

例えば、図９５に示す装置に新たなミドルウェアを追加

する場合を考える。ここで、一般的に、個々のフォーマットエンジンには独自の状態が規定されている。従って、フォーマットエンジンを管理する側（ミドルウェア読込部９５０２やミドルウェア実行部９５０３）は、フォーマットエンジン毎に異なる状態を把握している必要がある。すなわち、新たなミドルウェアが追加される場合、そのミドルウェアにおいて規定されている状態を理解することができるように、ミドルウェア読込部９５０２やミドルウェア実行部９５０３を変更しなければならない。以上より、新たなミドルウェアを追加する場合には、新たなミドルウェアをミドルウェア格納部９５０４に追加することに加えて、新たなミドルウェアに対応するようにミドルウェア読込部９５０２やミドルウェア実行部９５０３を変更しなければならない。従って、図９５に示すミドルウェア管理装置の設計の際にミドルウェアの追加や変更があると、開発者はミドルウェア読込部９５０２やミドルウェア実行部９５０３についても変更作業を行わなければならない。この変更作業は開発者にとって大きな負担になっていた。

それ故、本発明は、フォーマットエンジンの追加や変更に容易に対応することができる情報処理装置を提供することを目的とする。

上記目的を達成するために、本発明は、以下の構成を採用した。すなわち、本発明は、異なるフォーマットで記述されたデータをそれぞれ実行するためのフォーマットエンジンを格納している情報処理装置である。当該情報処理装置は、フォーマットエンジンの動作状態を全てのフォーマ

ットエンジンに共通の表現によって規定した共通状態を予め定義しておき、各フォーマットエンジンの動作を管理するフォーマットエンジン管理手段と、各フォーマットエンジンに対応して設けられ、共通状態と、フォーマットエンジンの動作状態を各フォーマットエンジン毎に異なる表現によって規定した動作状態である個別状態との対応を予め定義しておき、任意の個別状態となるようにフォーマットエンジンの動作を制御する動作制御手段とを備えている。フォーマットエンジン管理手段は、あるフォーマットエンジンを所定の共通状態に変化させる場合、当該所定の共通状態を示す共通状態情報を含むメッセージを、当該フォーマットエンジンに対応して設けられた動作制御手段へ送信する。各動作制御手段は、フォーマットエンジン管理手段からメッセージが送信されてきた場合、当該メッセージに含まれる共通状態情報により示される共通状態に対応する個別状態となるように、当該フォーマットエンジンを制御する。

また、情報処理装置は、各フォーマットエンジンに対応して設けられ、フォーマットエンジンの個別状態と、当該個別状態に対応する共通状態との組によって構成されるテーブル格納するテーブル格納手段をさらに備えていてもよい。このとき、各動作制御手段は、テーブルを参照することによって共通状態から個別状態を決定する。

また、情報処理装置は、各フォーマットエンジンに対応して設けられ、フォーマットエンジンの個別状態を取得し、取得した個別状態に対応する共通状態を示す共通状態情

報をフォーマットエンジン管理手段へ送信する個別状態取得手段をさらに備えていてもよい。このとき、フォーマットエンジン管理手段は、個別状態取得手段から出力された共通状態情報により示される共通状態に基づいて、各フォーマットエンジンの動作を管理する。

また、情報処理装置は、フォーマットエンジンが実行中に利用するリソースであって、複数のフォーマットエンジンが同時に利用することが不可能なリソースである極小リソースをさらに備えていてもよい。このとき、個別状態取得手段は、フォーマットエンジンから取得した個別状態が極小リソースを利用している動作状態を示す場合、当該フォーマットエンジンの共通状態情報として、所定の状態を示す共通状態情報をフォーマットエンジン管理手段へ出力する。また、個別状態取得手段は、フォーマットエンジンから取得した個別状態が極小リソースを利用していない動作状態を示す場合、当該フォーマットエンジンの共通状態情報として、所定の状態以外の状態を示す共通状態情報をフォーマットエンジン管理手段へ出力する。フォーマットエンジン管理手段は、共通状態情報が所定の状態を示すフォーマットエンジンが１つのみになるように、各フォーマットエンジンの動作を管理する。

また、フォーマットエンジン管理手段は、起動受付手段と、共通状態取得手段と、動作停止手段と、起動手段とを含んでいてもよい。起動受付手段は、フォーマットエンジンを起動するための起動要求を受け付ける。共通状態取得手段は、起動受付手段が起動要求を受け付けたことに応じ

て、状態取得手段から各フォーマットエンジンの共通状態情報を取得する。動作停止手段は、共通状態取得手段によって取得された共通状態情報が所定の状態を示すフォーマットエンジンがある場合、当該フォーマットエンジンの動作を停止させるメッセージを当該フォーマットエンジンに対応して設けられた動作制御手段へ送信する。起動手段は、動作停止手段によってフォーマットエンジンの動作が停止された後、起動要求に対応するフォーマットエンジンを起動させるメッセージを当該フォーマットエンジンに対応して設けられた動作制御手段へ送信する。

また、情報処理装置は、極小リソースと、リソース制御手段と、優先度情報格納手段とをさらに備えていてもよい。極小リソースは、フォーマットエンジンが実行中に利用するリソースであって、複数のフォーマットエンジンが同時に利用することが不可能なリソースである。リソース制御手段は、フォーマットエンジンの要求に応じてフォーマットエンジンに対してリソースの利用を許可する。優先度情報格納手段は、極小リソースを利用する場合における各フォーマットエンジン間の優先度を示す優先度情報を格納する。許可決定手段は、極小リソースを利用する要求が複数のフォーマットエンジン間で重複する場合、優先度情報に基づいて、当該極小リソースの利用を許可すべきフォーマットエンジンを決定する。このとき、リソース制御手段は、極小リソースを利用する要求が複数のフォーマットエンジン間で重複する場合、許可決定手段によって決定されたフォーマットエンジンのみに当該極小リソースの利用を

許可し、極小リソースを利用する要求が複数のフォーマットエンジン間で重複しない場合、当該要求を行ったフォーマットエンジンに当該極小リソースの利用を許可する。

また、極小リソースは複数設けられてもよい。このとき、リソース制御手段は極小リソースに対応して複数設けられる。

また、本発明は、異なるフォーマットで記述されたデータをそれぞれ実行するためのフォーマットエンジンを格納している情報処理装置のコンピュータで実行可能なプログラムの形態で提供されてもよい。

本発明によれば、フォーマットエンジン管理手段は各フォーマットエンジンの動作状態を共通状態で管理するので、フォーマットエンジンが追加または変更された場合であっても、フォーマットエンジン管理手段の構成を大幅に変更する必要がない。従って、フォーマットエンジンの追加や変更に容易に対応することができる。さらに、フォーマットエンジン管理部から動作制御部への通信は、共通状態情報を含むメッセージによって行われる。従って、フォーマットエンジン管理部は、フォーマットエンジンの違いを意識せずに各フォーマットエンジンの動作状態を管理できるので、複数のフォーマットエンジンの動作を同時に制御することが容易になる。

また、情報処理装置がテーブル格納手段をさらに備えることによって、個別状態と共通状態との相互の変換を容易に行うことができる。

また、さらに、情報処理装置が個別状態取得手段をさら

に備えることによって、フォーマットエンジン管理手段は、各フォーマットエンジンの動作状態を容易に取得することができる。

また、さらに、フォーマットエンジン管理手段が、共通状態情報が所定の状態を示すフォーマットエンジンが１つのみになるように、各フォーマットエンジンの動作を管理することによって、次の効果を得ることができる。すなわち、フォーマットエンジン管理手段は、極小リソースの利用が競合しないように、フォーマットエンジンの動作を管理することができる。換言すれば、フォーマットエンジン管理手段は、極小リソースの利用が競合しない範囲で、複数のフォーマットエンジンを同時に実行することも可能である。

また、さらに、フォーマットエンジン管理手段が、起動受付手段と、共通状態取得手段と、動作停止手段と、起動手段とを含んでいる場合には、次の効果を得ることができる。すなわち、フォーマットエンジン管理手段は、起動要求に対応するフォーマットエンジンを確実に起動させることができるとともに、極小リソースの利用が競合しない範囲で、起動中のフォーマットエンジンの実行を継続することができる。

また、さらに、情報処理装置が、極小リソースと、リソース制御手段と、優先度情報格納手段とをさらに備えている場合には、以下の効果を得ることができる。すなわち、フォーマットエンジン管理手段は、極小リソースの利用が競合しないように、フォーマットエンジンの動作を管理す

ることができる。換言すれば、フォーマットエンジン管理手段は、極小リソースの利用が競合しない範囲で、複数のフォーマットエンジンを同時に実行することも可能である。さらに、この場合、互いに異なる複数のフォーマットエンジンは、それぞれ、互いに異なる複数の極小リソースを同時に利用することも可能である。

また、極小リソースが複数設けられている場合には、各フォーマットエンジンは、それぞれ別個の極小リソースを同時に利用することができる。

図面の簡単な説明

図１は、本実施の形態に係る情報処理装置の一例であるデジタルテレビのハードウエア構成を表すブロック図である。

図２は、図１に示すデジタルテレビ１００の外観を示す図である。

図３は、入力部１１１を構成するフロントパネルの一例を示す図である。

図４は、第１の動作例を行う場合における、図１に示すデジタルテレビの機能的な構成を示すブロック図である。

図５は、テーブル格納部４４が格納するテーブルの一例を示す図である。

図６は、第２の動作例を行う場合における、図１に示すデジタルテレビの機能的な構成を示すブロック図である。

図７は、情報処理装置の機能的な構成をより具体的に示す図である。

図８は、ディスプレイ１０７に表示されるフォーマットエンジンの一覧の例を示す図である。

図９は、ディスプレイ１０７に表示されるフォーマットエンジンの一覧の例を示す図である。

図１０は、ディスプレイ１０７に表示されるフォーマットエンジンの一覧の例を示す図である。

図１１は、ディスプレイ１０７に表示される、実行可能なアプリケーションの一覧の例を示す図である。

図１２は、ディスプレイ１０７に表示される、実行可能なアプリケーションの一覧の例を示す図である。

図１３は、ディスプレイ１０７に表示される、実行可能なアプリケーションの一覧の他の例を示す図である。

図１４は、ディスプレイ１０７に表示される、実行可能なアプリケーションの一覧の他の例を示す図である。

図１５は、ディスプレイ１０７に表示される、実行可能なＨＴＭＬデータの一覧の例を示す図である。

図１６は、ディスプレイ１０７に表示される、実行可能なＨＴＭＬデータの一覧の例を示す図である。

図１７は、メッセージフォーマットの一例を示す図である。

図１８は、サブプログラムＩＤの表の一例を示す図である。

図１９は、Ｍｅｓｓａｇｅ　ＩＤフィールド１７３に格納されるメッセージＩＤの表の一例を示す図である。

図２０は、フォーマットエンジン状態応答メッセージにおけるデータフィールド１７５のフォーマットの一例を示

す図である。

図２１は、フォーマットエンジンの動作状態と動作状態ＩＤとの対応を示す表の一例を示す図である。

図２２は、アプリケーション・データ一覧応答メッセージにおけるデータフィールド１７５のフォーマットの一例を示す図である。

図２３は、「アプリケーション・データ実行」「アプリケーション・データ停止」「アプリケーション・データ一時停止」のメッセージにおけるデータフィールド１７５のフォーマットの一例を示す図である。

図２４は、「フォーマットエンジン状態要求」メッセージの具体例を示す図である。

図２５は、「フォーマットエンジン状態要求」メッセージの具体例を示す図である。

図２６は、「フォーマットエンジン状態要求」メッセージの具体例を示す図である。

図２７は、「フォーマットエンジン取得応答」メッセージの具体例を示す図である。

図２８は、「フォーマットエンジン取得応答」メッセージの具体例を示す図である。

図２９は、「フォーマットエンジン取得応答」メッセージの具体例を示す図である。

図３０は、アプリケーション・データ一覧要求」メッセージの具体例を示す図である。

図３１は、「アプリケーション・データ取得応答」メッセージの具体例を示す図である。

図３２は、状態管理部７３２が保持している情報の一例を示す図である。

図３３は、状態管理部７３２が保持している情報の一例を示す図である。

図３４は、状態管理部７３２が保持している情報の一例を示す図である。

図３５は、「フォーマットエンジン状態変化」メッセージの一例を示す図である。

図３６は、状態管理部７３２が送信するメッセージの一例を示す図である。

図３７は、メーラー７６０が送信するメッセージの一例を示す図である。

図３８は、図３２の変化後の状態を示す図である。

図３９は、リソース管理部７３３の内部構成を示すブロック図である。

図４０は、プロセス記憶部３９０１において記憶されている情報の一例を示す図である。

図４１は、優先度記憶部３９０２に記憶されている情報の一例を示す図である。

図４２は、最新起動記憶部３９０３が記憶している情報の一例を示す図である。

図４３は、リソースＩＤ記憶部３９０４が記憶している情報の一例を示す図である。

図４４は、リソースＩＤの定義をプログラミング言語であるＣ言語で記述した例を示す図である。

図４５は、Ｊａｖａミドルウェア７４０の構成を示すブ

ロック図である。

図４６は、ＡＩＴの主要部を表した模式図である。

図４７は、Ｊａｖａアプリケーションの４つの動作状態および動作状態間の遷移を表す状態遷移図である。

図４８は、ＡＩＴの主要部を表した模式図である。

図４９は、、Ｊａｖａミドルウェア７４０が実行可能な全てのＪａｖａアプリケーションの動作状態と、Ｊａｖａミドルウェア７４０の動作状態との対応を示す変換表の一例を示す図である。

図５０は、各Ｊａｖａアプリケーションの動作状態を共通状態に変換するための変換表の一例を示す図である。

図５１は、第１の変換部７４１が送信するメッセージの具体例を示す図である。

図５２は、ＨＴＭＬブラウザ７５０の構成を示すブロック図である。

図５３は、ＤＶＢ－ＨＴＭＬデータの一例を示す図である。

図５４は、ＶＢ－ＨＴＭＬデータのレイアウトを決めるスタイルシートの一例を示す図である。

図５５は、ＤＯＭツリーの例を示す図である。

図５６は、図５５に示すＤＯＭツリーを描画した際のディスプレイ１０７を示す図である。

図５７は、ＤＶＢ－ＨＴＭＬデータの例を示す図である。

図５８は、ＤＶＢ－ＨＴＭＬデータの例を示す図である。

図５９は、図５８に示すＤＶＢ－ＨＴＭＬデータに対してパーザー５２０１が構築したＤＯＭツリーの一部を示す図である。

図６０は、レイアウター５２０２および描画部５２０３によって表示されたディスプレイ表示の例を示す図である。

図-６１は、ＤＯＭツリーの変更結果を示す図である。

図６２は、ディスプレイ表示の変更結果の例を示す図である。

図６３は、ＡＩＴの主要部を表した模式図である。

図６４は、ＤＶＢ－ＨＴＭＬデータの５つの状態および状態間の遷移を表す状態遷移図である。

図６５は、ＡＩＴの主要部を表した模式図である。

図６６は、ＤＶＢ－ＨＴＭＬデータ内に他のＤＶＢ－ＨＴＭＬデータへのリンクが定義されている例を示す図である。

図６７は、図６６に示すＤＶＢ－ＨＴＭＬデータの表示例を示す図である。

図６８は、ＤＶＢ－ＨＴＭＬデータの表示例を示す図である。

図６９は、ＨＴＭＬブラウザ７５０が表示可能な全てのＨＴＭＬデータの状態と、ＨＴＭＬブラウザ７５０の動作状態との対応を示す変換表の一例を示す図である。

図７０は、各ＨＴＭＬデータの状態を共通状態に変換するための変換表の一例を示す図である。

図７１は、第２の変換部７５１が送信するメッセージの

具体例を示す図である。

図７２は、メーラー７６０の状態と、結合部７３０が規定している共通状態との対応を示す変換表の一例を示す図である。

図７３は、第３の変換部７６１が生成する「アプリケーション・データ一覧応答」メッセージの具体例を示す図である。

図７４は、第１の動作例の動作を行う場合におけるデジタルテレビにおける処理の流れを示すフローチャートである。

図７５は、通信部７３１が行うメッセージ送信の処理の流れを示すフローチャートである。

図７６は、第１の変換部７４１が「フォーマットエンジン状態要求」メッセージを受け取ったときの処理の流れを示すフローチャートである。

図７７は、第３の変換部７７１が「フォーマットエンジン状態要求」メッセージを受け取ったときの処理の流れを示すフローチャートである。

図７８は、第１の変換部７４１が、「フォーマットエンジン停止」メッセージを受け取った際の処理の流れを表すフローチャートである。

図７９は、第２の変換部７５１が、「フォーマットエンジン停止」メッセージを受け取った際の処理の流れを示すフローチャートである。

図８０は、第３の変換部７６１が、「フォーマットエンジン停止」メッセージを受け取った際の処理の流れを示す

フローチャートである。

図８１は、第１の変換部７４１が、「フォーマットエンジン実行」メッセージを受け取った際の処理の流れを示すフローチャートである。

図８２は、第１の変換部７４１が、「フォーマットエンジン一時停止」メッセージを受け取った際の処理の流れを表すフローチャートである。

図８３は、第２の動作例の動作を行う場合におけるデジタルテレビにおける処理の流れを示すフローチャートである。

図８４は、リソース剥奪通知を受け取った変換部における処理の流れを示すフローチャートである。

図８５は、第１の変換部７４１がアプリケーションマネージャ４５０３からＪａｖａアプリケーションの状態変化通知を受けた際の処理の流れを示すフローチャートである。

図８６は、第１の変換部７４１が「アプリケーション・データ一覧要求」メッセージを受け取った際の処理の流れを示すフローチャートである。

図８７は、第１の変換部７４１が、「アプリケーション・データ実行」メッセージを受け取った際の処理の流れを示すフローチャートである。

図８８は、第１の変換部７４１が、「アプリケーション・データ停止」メッセージを受け取った際の処理の流れを示すフローチャートである。

図８９は、第１の変換部７４１が、「アプリケーション

・データー時停止」メッセージを受け取った際の処理の流れを示すフローチャートである。

図９０は、第３の変換部７６１がメーラー７６０が終了したことの通知をメーラー７６０から受けた際の処理の流れを示すフローチャートである。

図９１は、結合部７３０が「フォーマットエンジン状態変化」メッセージを受け取った場合の結合部７３０の状態管理部７３２の処理の流れを示すフローチャートである。

図９２は、結合部７３０が「アプリケーション・データ一覧変化」メッセージを受け取った場合の結合部７３０の状態管理部７３２の処理の流れを示すフローチャートである。

図９３は、フォーマットエンジンを実行しても、他のフォーマットエンジンに大きな影響を与えない状況の組み合わせを示す図である。

図９４は、従来のデジタル放送受信機の構成図の一例である。

図９５は、従来のデジタルテレビにおいて用いられるミドルウェア切替装置の構成図である。

発明を実施するための最良の形態

以下、本発明に係る情報処理装置の実施の形態を、図面を参照しながら説明する。なお、以下の実施の形態では、情報処理装置の一例としてデジタルテレビを用いて説明するが、本発明は携帯電話や携帯情報端末等においても適用可能である。

（ハードウェア構成）

まず、図１～図３を参照して、情報処理装置のハードウェア構成について説明する。図１は、本実施の形態に係る情報処理装置の一例であるデジタルテレビのハードウエア構成を表すブロック図である。図１において、デジタルテレビ１００は、チューナ１０１、デスクランブラ１０２、ＴＳデコーダ１０３、オーディオデコーダ１０４、スピーカ１０５、ビデオデコーダ１０６、ディスプレイ１０７、２次記憶部１０８、１次記憶部１０９、ＲＯＭ１１０、入力部１１１、ネットワークインターフェース１１２、およびＣＰＵ１１３を備えている。

図２は、図１に示すデジタルテレビ１００の外観を示す図である。図１に示すデジタルテレビ１００は、例えば図２に示すような薄型テレビとして実現される。図２に示すように、デジタルテレビ１００は、図１に示す構成に加え、筐体２０１および信号入力端子２０４をさらに備えている。なお、ディスプレイ２０２は、図１に示すディスプレイ１０７に相当する。また、フロントパネル部２０３は、図１に示す入力部１１１に相当する。フロントパネル部２０３は、複数のボタンで構成される。なお、フロントパネル部２０３の詳細な構成は後述する図３に示されている。信号入力端子２０４は、地上波放送の放送局、衛星のアンテナ、あるいはケーブルテレビの局システムからの放送信号をデジタルテレビ１００に入力するための端子である。信号入力端子２０４は、図１のチューナ１０１と接続されている。なお、デジタルテレビ１００は、インターネット

に接続するためのネットワーク端子を有している。ネットワーク端子は、ネットワークインターフェース１１２と接続されている。

次に、デジタルテレビ１００の基本的な動作を説明する。チューナ１０１は、信号入力端子２０４を介して送信されてきた放送信号を入力する。チューナ１０１は、ＣＰＵ１１３によって指定された周波数を示すチューニング情報に従って、入力した放送信号を復調する。復調された放送信号は、デスクランブラ１０２に引き渡される。デスクランブラ１０２は、復調された放送信号を復号する。復号に必要な鍵はＣＰＵ１１３から与えられる。復号によって得られるＭＰＥＧ２トランスポートストリームは、ＴＳデコーダ１０３へ出力される。

ＴＳデコーダ１０３は、デスクランブラ１０２から受け取ったＭＰＥＧ２トランスポートストリームに対してフィルタリングを行い、必要なデータをオーディオデコーダ１０４、ビデオデコーダ１０６あるいはＣＰＵ１１３に引き渡す。ＭＰＥＧ２トランスポートストリームの詳細はＭＰＥＧ規格書“ＩＳＯ／ＩＥＣ１３８１８－１”に記載されているので詳細な記述は省略する。ＭＰＥＧ２トランスポートストリームは、複数の固定長パケットで構成され、各パケットには映像、音声、字幕データ、およびアプリケーション等が格納されている。また、各パケットにはパケットＩＤが与えられている。ＴＳデコーダ１０３は、ＣＰＵ１１３によってパケットＩＤとそのパケットＩＤを有するパケットの出力先の組を指示される。パケットの出力先は

「オーディオデコーダ１０４」、「ビデオデコーダ１０６」、および「ＣＰＵ１１３」のいずれかである。例えば、ＣＰＵ１１３が、パケットＩＤが「１２３」であるパケットの出力先が「ビデオデコーダ１０６」である旨の指示をＴＳデコーダ１０３へ出力した場合、ＴＳデコーダ１０３は、ＭＰＥＧ２トランスポートストリームからパケットＩＤが「１２３」であるパケットを抽出し、そのパケットをビデオデコーダ１０６に引き渡す。なお、ＴＳデコーダは、複数のフィルタリング処理を変更して実行することができる。

オーディオデコーダ１０４は、ＴＳデコーダ１０３から引き渡された各パケットに埋め込まれているオーディオデータを連結し、デジタルーアナログ変換を行いスピーカ１０５へ出力する。スピーカ１０５は、オーディオデコーダ１０４から出力された信号を音声出力する。ビデオデコーダ１０６は、ＴＳデコーダ１０３から引き渡された各パケットに埋め込まれているビデオデータを連結し、デジタルーアナログ変換を行うことによって得られるビデオ信号をディスプレイ１０７へ出力する。ディスプレイ１０７は、典型的にはブラウン管や液晶表示装置等で構成され、ビデオデコーダ１０６から出力されたビデオ信号を出力する。さらに、ディスプレイ１０７は、ＣＰＵ１１３から指示されたメッセージ等を表示することもある。

２次記憶部１０８は、典型的にはフラッシュメモリやハードディスク等で構成され、ＣＰＵ１１３から指示されたデータやプログラムを保存する。保存されているデータや

プログラムはＣＰＵ１１３に参照される。保存されているデータやプログラムは、デジタルテレビ１００の電源が切断された状態でも保存しつづける。１次記憶部１０９は、典型的にはＲＡＭ等で構成され、ＣＰＵ１１３から指示されたデータやプログラムを一次的に保存する。保存されているデータやプログラムはＣＰＵ１１３に参照される。保存されているデータやプログラムは、デジタルテレビ１００の電源が切断された際に、抹消される。ＲＯＭ１１０は、書き換え不可能なメモリデバイスであり、典型的にはＲＯＭやＣＤ－ＲＯＭ、ＤＶＤ等で構成される。ＲＯＭ１１０には、ＣＰＵ１１３によって実行されるプログラムが格納されている。

入力部１１１は、典型的にはフロントパネルやリモコンで構成され、ユーザからの入力を受け付ける。図３は、入力部１１１を構成するフロントパネルの一例を示す図である。図３において、フロントパネル３００は７つのボタン、すなわち、上カーソルボタン３０１、下カーソルボタン３０２、左カーソルボタン３０３、右カーソルボタン３０４、ＯＫボタン３０５、取消ボタン３０６、ＭＥＮＵボタン３０７、Ｃ＋ボタン３０８、およびＣ－ボタン３０９を備えている。ユーザが各ボタンを押下すると、押下されたボタンを示す識別子がＣＰＵ１１３に通知される。

ネットワークインターフェース１１２は、典型的にはモデムやイーサネットコネクター等で構成され、インターネットやイントラネット等のネットワークと接続する機能をデジタルテレビ１００に提供するものである。ネットワー

クインターフェース１１２は、ＣＰＵ１１３からの指示に従い、ネットワーク上に接続された他の情報機器とのデータの送受信を行う。例えばネットワークインターフェース１１２がモデムによって構成されている場合、ネットワークインターフェース１１２は、ＣＰＵ１１３から接続先の電話番号や通信パラメータ情報を受け取り、ネットワークとの通信経路を確立する。

ＣＰＵ１１３は、ＲＯＭ１１０が記憶するプログラムを実行する。ＣＰＵ１１３は、プログラムの指示に従い、チューナ１０１、デスクランブラ１０２、ＴＳデコーダ１０３、ディスプレイ１０７、２次記憶部１０８、１次記憶部１０９、およびＲＯＭ１１０の動作を制御する。ＣＰＵ１１３の詳細な動作については後述する。

（動作概要）

次に、情報処理装置の詳細な構成および動作を説明する前に、図４～図６を参照して、本発明の目的を達成するための情報処理装置の動作の概要を説明する。ここでは、情報処理装置の動作例として、第１および第２の動作例という２つの動作例について説明する。

（第１の動作例）

まず、図４および図５を用いて第１の動作例の概要を説明する。図４は、第１の動作例を行う場合における、図１に示すデジタルテレビの機能的な構成を示すブロック図である。図４において、デジタルテレビは、フォーマットエンジン管理部４１と、第１の変換部４２と、第２の変換部４３と、第１のテーブル格納部４４と、第２のテーブル格

納部４９と、第１のフォーマットエンジン４５と、第２のフォーマットエンジン４６と、リソース制御部４７と、リソース４８とを備えている。図４において、リソース４８以外の各構成要素は、図１に示すＣＰＵ１１３がＲＯＭ１１０に格納されたプログラムを１次記憶部１０９および２次記憶部１０８を用いて実行することによって実現される。また、リソース４８は、フォーマットエンジンによって利用されるハードウェアである。リソース４８は、例えば、メモリ（１次記憶部１０９および２次記憶部１０８）や、チューナ１０１、ＴＳデコーダ１０３、ネットワークインターフェース１１２等である。なお、図４では、説明を容易にする目的で、デジタルテレビ１００が有するフォーマットエンジンの数を２つとして説明するが、フォーマットエンジンの数は２つ以上であってもよい。

ここで、各フォーマットエンジン４５および４６は、複数種類の動作状態を取り得る。すなわち、各フォーマットエンジン４５および４６は、「実行されている状態」や、「実行されていない状態」等の状態を取り得る。以下、情報処理装置の動作概要を説明する図４～図６においては、第１のフォーマットエンジン４５は、「動作中」、「停止中」、および「一時停止中」と表現される３種類の動作状態を取り得るものとする。また、第２のフォーマットエンジン４６は、「起動中」および「起動中でない」という２種類の動作状態を取り得るものとする。ここで、各フォーマットエンジンにおいて個別に規定された動作状態を、個別状態と呼ぶ。つまり、第１のフォーマットエンジン４５

においては、「動作中」、「停止中」、および「一時停止中」と表現される３種類の動作状態を含む個別状態が規定されている。また、第２のフォーマットエンジン４６においては、「起動中」および「起動中でない」という２種類の動作状態を含む個別状態が規定されている。なお、一般的に、各フォーマットエンジンが取り得る動作状態の種類は、フォーマットエンジンによって様々である。

フォーマットエンジン管理部４１は、各フォーマットエンジン４５および４６の動作を管理する。具体的には、各フォーマットエンジン４５および４６に対して、起動や停止等の指示を行う。ここで、フォーマットエンジン管理部４１においては、各フォーマットエンジンが取り得る動作状態は、共通状態として予め規定されている。共通状態とは、全てのフォーマットエンジンに共通の表現によってフォーマットエンジンの動作状態を規定した概念である。図４においては、フォーマットエンジン管理部４１においては、「実行中」および「終了」と表現される２種類の動作状態を含む共通状態が規定されるものとする。このように、共通状態は、各フォーマットエンジン４５および４６において規定されている動作状態（すなわち、個別状態）とは異なる表現であってもよい。また、共通状態に含まれる動作状態の種類は２種類に限らない。共通状態に含まれる動作状態の種類は、フォーマットエンジン管理部４１が各フォーマットエンジンを管理するために必要となる数の動作状態の種類を定義しておけばよい。

また、フォーマットエンジン管理部４１は、例えばユー

ザからの入力に従って、フォーマットエンジンに対して動作を指示する。このとき、フォーマットエンジン管理部４１は、共通状態情報を用いて指示を行う。共通状態情報とは、フォーマットエンジンの動作状態を共通状態によって表現した情報である。図４において、例えば、フォーマットエンジン管理部４１が第１のフォーマットエンジン４５の動作状態が所定の動作状態となるように指示を行う場合、フォーマットエンジン管理部４１は、当該所定の動作状態を示す共通状態情報を第１の変換部４２へ出力する。これと同様に、フォーマットエンジン管理部４１は、第２のフォーマットエンジン４６に対して動作の指示を行う場合、第２の変換部４３へ共通状態情報を出力する。

また、フォーマットエンジン管理部４１は、各変換部４２および４３から出力されてくる共通状態情報を受け取ることによって、各フォーマットエンジン４５および４６の動作状態を共通状態の表現によって把握する。このように把握された共通状態を用いて、フォーマットエンジン管理部４１は、各フォーマットエンジン４５および４６の動作を管理することができる。

第１の変換部４２は、フォーマットエンジン管理部４１から出力されてくる共通状態情報に基づいて第１のフォーマットエンジン４５の動作を制御する。すなわち、第１の変換部４２は、フォーマットエンジン管理部４１において規定される共通状態と、第１のフォーマットエンジン４５において規定される個別状態との対応を予め規定しておく。フォーマットエンジン管理部４１から共通状態情報が出

力されてくると、第１の変換部４２は、当該対応に基づいて、共通状態情報により示される共通状態に対応する個別状態を特定する。さらに、第１の変換部４２は、特定した個別状態となるように第１のフォーマットエンジン４５の動作を制御する。

また、第１の変換部４２は、第１のフォーマットエンジン４５の動作状態（個別状態で表現されている）を取得し、取得した動作状態を共通状態によって表現される形でフォーマットエンジン管理部４１へ通知する。つまり、第１の変換部４２は、第１のフォーマットエンジン４５から取得した動作状態を示す共通状態情報をフォーマットエンジン管理部４１へ出力する。共通状態情報をフォーマットエンジン管理部４１へ出力する場合も、上記共通状態と個別状態との対応が用いられる。すなわち、第１のフォーマットエンジン４５から個別状態を取得すると、第１の変換部４２は、上記対応に基づいて、取得した個別状態に対応する共通状態を特定する。さらに、第１の変換部４２は、特定した共通状態を示す共通状態情報をフォーマットエンジン管理部４１へ出力する。

第２の変換部４３は、第１の変換部４２と同様の処理を行う。すなわち、第２の変換部４３は、フォーマットエンジン管理部４１から出力されてくる共通状態情報に基づいて第２のフォーマットエンジン４６の動作を制御する。また、第２の変換部４３は、第２のフォーマットエンジン４６の動作状態（個別状態で表現されている）を取得し、取得した動作状態を共通状態によって表現される形でフォー

マットエンジン管理部４１へ通知する。これらの動作の詳細は、第１の変換部４２における動作と同様の方法で行われる。

第１のテーブル格納部４４は、フォーマットエンジン管理部４１において規定される共通状態と、第１のフォーマットエンジン４５において規定される個別状態とを対応付けたテーブルを格納する。また、第２のテーブル格納部４４は、フォーマットエンジン管理部４１において規定される共通状態と、第２のフォーマットエンジン４６において規定される個別状態とを対応付けたテーブルを格納する。図５は、各テーブル格納部４４および４９が格納するテーブルの一例を示す図である。図５（ａ）は、第１のテーブル格納部４４が格納するテーブルの一例を示す図であり、図５（ｂ）は、第２のテーブル格納部４９が格納するテーブルの一例を示す図である。図５（ａ）においては、「実行中」という共通状態に対して、「動作中」という個別状態が対応付けられる。また、「終了」という共通状態に対して、「停止中」および「一時停止中」という個別状態が対応付けられる。図５（ｂ）においては、「実行中」という共通状態に対して、「起動中」という個別状態が対応付けられている。また、「終了」という共通状態に対して、「起動中でない」という個別状態が対応付けられている。各変換部４２および４３は、図５に示すテーブルを参照して、フォーマットエンジン管理部４１から出力されてくる共通状態情報により示される共通状態に対応する個別状態を特定する。また、各変換部４２および４３は、図５に示

すテーブルを参照して、各フォーマットエンジン４５および４６から取得した個別状態に対応する共通状態を特定する。

例えば、第１の変換部４５に「実行中」を示す共通状態情報が出力されてきた場合、第１の変換部４５は、図５（ａ）に示すテーブルを参照して、「動作中」という個別状態を特定する。そして、第１の変換部４５は、特定した「動作中」という動作状態となるように第１のフォーマットエンジン４５を制御する。また、例えば、第２の変換部４３が第２のフォーマットエンジン４６から「起動中でない」を示す個別状態を取得した場合、第２の変換部４３は、上記図５（ｂ）に示すテーブルを参照して、「終了」という共通状態を特定する。そして、第２の変換部４３は、特定した「終了」を示す共通状態情報をフォーマットエンジン管理部４１へ出力する。

また、図４において、リソース制御部４７は、各フォーマットエンジン４５および４６の要求に応じて、各フォーマットエンジン４５および４６に対してリソース４８の利用を許可する。

なお、以上の動作は、後述する第２の動作例においても同様である。

ここで、リソース４８が極小リソースである場合を考える。極小リソースとは、フォーマットエンジンが実行中に利用するリソースであって、複数のフォーマットエンジンが同時に利用することが不可能なリソースである。例えば、チューナ１０１や、ＴＳデコーダ１０３や、ネットワー

クインターフェース１１２は極小リソースである。極小リソースは、デジタルテレビ１００が有している数が限られたリソースで、かつ、複数のフォーマットエンジンが同時に使用を試み、競合する可能性のあるリソースのことである。例えば、チューナ１０１は１つの周波数帯にしかチューニングできない。複数のフォーマットエンジンが異なる周波数帯にチューニングしたい場合にリソースの競合が発生する。このような極小リソースは複数のフォーマットエンジンが同時に利用することができないので、リソース制御部４７は、複数のフォーマットエンジンに対して極小リソースの利用を同時に許可しないようにする必要がある。もしリソース制御部４７が複数のフォーマットエンジンに対して極小リソースの利用を同時に許可すれば、当該フォーマットエンジンの動作が正常に行われない可能性がある。そこで、図４においては、フォーマットエンジンの動作を正常に行うための方法として、以下の方法をとる。

図４においては、各テーブル格納部４４および４９に格納されているテーブルにおける共通状態と個別状態との対応付けは、次のように行われる。すなわち、フォーマットエンジンが極小リソースを利用する動作状態を表す個別状態に対して、「実行中」という共通状態が対応付けられる。さらに、フォーマットエンジンが極小リソースを利用しない動作状態を表す個別状態に対して、「終了」という共通状態が対応付けられる。なお、図４においては、第１のフォーマットエンジン４５は、「動作中」という個別状態においてのみ極小リソースを利用し、他の個別状態におい

ては極小リソースを利用しないものとする。また、第２のフォーマットエンジン４６は、「起動中」という個別状態においてのみ極小リソースを利用し、他の個別状態においては極小リソースを利用しないものとする。このとき、上記テーブルは図５に示すようになる。

上記テーブルの対応付けを以上のように行うとともに、さらに、フォーマットエンジン管理部４１は、共通状態が「実行中」であるフォーマットエンジンが１つになるように、各フォーマットエンジン４５および４６の動作を管理する。これによって、フォーマットエンジン管理部４１は、極小リソースを利用するフォーマットエンジンが１つになるように各フォーマットエンジン４５および４６の動作を管理することができる。フォーマットエンジン管理部４１による具体的な管理方法は種々考えられる。具体的な管理方法については、後述する処理例において述べる。

（第２の動作例）

次に、図６を用いて第２の動作例について説明する。図６は、第２の動作例を行う場合における、図１に示すデジタルテレビの機能的な構成を示すブロック図である。図６において、デジタルテレビは、図４に示す構成に加えて、許可決定部６１と、優先度情報格納部６２とを備えている。なお、図６において、図４に示す構成要素と同じ構成要素には同じ参照符号を付している。図６において、許可決定部６１および優先度情報格納部６２は、図１に示すＣＰＵ１１３がＲＯＭ１１０に格納されたプログラムを１次記憶部１０９および２次記憶部１０８を用いて実行すること

によって実現される。

図４における動作と図６における動作との相違点は、リソース４８が極小リソースである場合における、フォーマットエンジンの動作を正常に行うための方法である。その他の動作、例えば、フォーマットエンジン管理部４１が共通状態によって各フォーマットエンジン４５および４６の動作を管理する動作や、各変換部４２および４３が共通状態と個別状態との相互の変換を行う動作については、図４と同様である。従って、以下では、第２の動作例においてフォーマットエンジンの動作を正常に行うための方法を説明する。

第２の動作例では、各テーブル格納部４４および４９において、フォーマットエンジンが極小リソースを利用する動作状態を表す個別状態に対して、「実行中」という共通状態が対応付けられる必要はない。さらに、フォーマットエンジン管理部４１は、共通状態が「実行中」であるフォーマットエンジンが１つになるように、各フォーマットエンジン４５および４６の動作を管理する必要はない。第２の動作例では、リソース４８を利用する要求がリソース制御部４７において重複した場合、リソースを使用可能なフォーマットエンジンが優先度情報に基づいて選択される。以下、詳細を説明する。

優先度情報格納部６２は、優先度情報を格納する。優先度情報とは、極小リソースであるリソース４８を利用する場合における各フォーマットエンジン間の優先度を示す情報である。図６においては、優先度情報は、第１のフォー

マットエンジン４５と第２のフォーマットエンジン４６とのいずれを優先するかを示す情報である。

リソース制御部４７は、リソース４８を利用する要求がリソース制御部４７において重複した場合、すなわち、リソース４８を利用する要求が複数のフォーマットエンジンから同時に行われた場合、要求が重複した旨の通知を許可決定部６１へ出力する。

許可決定部６１は、要求が重複した旨の通知をリソース制御部４７から受け取った場合、優先度情報格納部６２に格納されている優先度情報に基づいて、リソース４８の利用を許可すべきフォーマットエンジンを決定する。さらに、許可決定部６１は、決定したフォーマットエンジンを示す情報をリソース制御部４７へ出力する。リソース制御部４７は、当該情報により示されるフォーマットエンジンにリソース４８の利用を許可する。以上によって、極小リソースを利用するフォーマットエンジンが１つになるように各フォーマットエンジン４５および４６の動作を管理することができる。なお、利用が許可されなかったフォーマットエンジンに対して、利用を許可しない旨の通知がリソース制御部４７またはフォーマットエンジン管理部４１によって行われてもよい。

以上のように、上記第１および第２動作例によれば、フォーマットエンジン管理部４１は共通状態を用いて各フォーマットエンジン４５および４６の動作を管理する。共通状態は各フォーマットエンジンにおいて規定される個別状態とは別個であり、管理すべきフォーマットエンジンとは

無関係である。従って、仮にフォーマットエンジンの追加や変更があった場合でも、共通状態として規定されている内容を変更する必要はない。つまり、フォーマットエンジン管理部４１を変更する必要はない。フォーマットエンジンの追加があった場合には、当該フォーマットエンジンを追加するとともに、それに対応する変換部を追加すればよいだけである。つまり、追加するフォーマットエンジンにおいて規定される個別状態と共通状態との対応を新たに追加すればよいだけである。以上のように、上記第１および第２動作例によれば、仮にフォーマットエンジンの追加や変更があった場合でも、当該追加や変更に容易に対応することができる。

さらに、リソース４８が極小リソースである場合、上記第１および第２動作例によれば、極小リソースを複数のフォーマットエンジンが同時に利用しないように各フォーマットエンジンの動作が管理される。従って、極小リソースを利用する複数のフォーマットエンジンが正常に動作しなくなることを防止することができる。

なお、上記第１および第２の動作例においては、リソース４８が１つである場合について説明したが、リソースは複数であってもよい。リソースが複数である場合、リソース制御部はリソースに対応して、リソースと同数設けられる必要がある。なお、第１の動作例においてリソースが複数である場合であっても、フォーマットエンジンが極小リソースを利用する動作状態を表す個別状態に対して、「実行中」という共通状態が対応付けられる点は上記第１の動

作例と同様である。また、第１の動作例においてリソースが複数である場合であっても、フォーマットエンジン管理部４１が、共通状態が「実行中」であるフォーマットエンジンが１つになるように、各フォーマットエンジン４５および４６の動作を管理する点は上記第１の動作例と同様である。

また、第２の動作例においてリソースが複数である場合、極小リソースに対応して設けられるリソース制御部は、上記第２の動作例と同様の動作を行う。これによって、極小リソースが複数である場合、各極小リソースが重複して利用されることを防止することができる。さらに、第２の動作例では、互いに異なる複数のフォーマットエンジンは、それぞれ、互いに異なる複数の極小リソースを同時に利用することが可能である。つまり、あるフォーマットエンジンＡが極小リソースＡを利用している場合に、別のフォーマットエンジンＢは極小リソースＢを利用することができる。なお、第１の動作例では、極小リソースが複数である場合、複数の極小リソースを同時に利用することは不可能であった。これに対して第２の動作例では、複数の極小リソースを同時に利用することができる点で、第１の動作例よりも効率よくフォーマットエンジンを実行することができると言える。

なお、上記第１および第２の動作例においては、各変換部４５および４６は、フォーマットエンジンと別個に構成されるものとしたが、フォーマットエンジンと一体的に構成されるものであってもよい。換言すれば、フォーマット

エンジンは、変換部の機能を有するものであってもよい。以上で、第１および第２の動作例の概要の説明を終了する。

（具体的な構成例）

次に、図７～図７３を用いて、本発明に係る情報処理装置の具体的な構成を説明する。なお、以下で説明する構成は、上述した第１および第２の動作例の双方を実現することが可能な構成である。従って、以下で説明する構成には、第１の動作例のみを実行する場合には不要である構成や、第２の動作例のみを実行する場合には不要である構成が含まれるが、そのような構成については説明中において適宜述べる。

図７は、情報処理装置の機能的な構成をより具体的に示す図である。図７に示す各構成要素は、図１に示すＣＰＵ１１３がＲＯＭ１１０に格納されたプログラムを１次記憶部１０９および２次記憶部１０８を用いて実行することによって実現される。換言すれば、図７は、ＲＯＭ１１０に記憶され、ＣＰＵ１１３に実行されるプログラムの構成を示す図である。

図７において、情報処理装置は、ＯＳ７１０と、ナビゲータ７２０と、結合部７３０と、Ｊａｖａミドルウェア７４０と、ＨＴＭＬブラウザ４５０と、メーラー４６０と、第１の変換部７４１と、第２変換部４５１と、第３の変換部７６１とを備えている。これらは、ＲＯＭ１１０に格納されているプログラムのサブプログラムである。なお、図４に示すフォーマットエンジン管理部４１は、ナビゲータ

７２０および結合部７３０の機能の一部として実現される。

図７に示すサブプログラムの内、Ｊａｖａミドルウェア７４０、ＨＴＭＬブラウザ７５０、およびメーラー７６０は、所定のフォーマットで記述されたアプリケーションの実行や、所定のフォーマットで記述されたデータの表示等を行うフォーマットエンジンである。以下に説明する具体的な構成例では、これら３種類のフォーマットエンジンを例として取り上げたが、フォーマットエンジンは、ワードプロセッサプログラムやスプレッドシートプログラム等であってもよい。また、情報処理装置が備えるフォーマットエンジンの数は３つに限るものではなく、１つでも２つでもよく、４つ以上でもよい。ただし、フォーマットエンジンの数と同数の変換部を用意する必要がある。

ＯＳ７１０は、デジタルテレビ１００の電源が投入されると、ＣＰＵ１１３が起動するサブプログラムである。ＯＳ７１０は、オペレーティングシステムの略であり、Ｌｉｎｕｘ等が一例である。ＯＳ７１０は、他のサブプログラムを平行して実行するカーネル７１１およびライブラリ７１２等で構成されるプログラム群の総称であり、詳細な説明は省略する。本構成例においては、ＯＳ７１０のカーネル７１１は、電源投入後にナビゲータ７２０、Ｊａｖａミドルウェア７４０、ＨＴＭＬブラウザ７５０、およびメーラー７６０をサブプログラムとして起動する。ＯＳ７１０がＬｉｎｕｘである場合、ＯＳ７１０は、これらサブプログラムに対してそれぞれ個別のプロセスを起動し、それぞ

れのプロセス内で個々のサブプログラムを実行する。

ＯＳ７１０のライブラリ７１２は、他のサブプログラムに対して、デジタルテレビ１００が有するハードウエア構成要素（リソース）を制御するための複数の機能を提供する。つまり、図４に示すリソース制御部４７は、ライブラリ７１２の機能の一部として実現される。本構成例では、複数の機能の一例として、チューナライブラリ（チューナＬｉｂ）７１２１、デスクランブラライブラリ（デスクランブラＬｉｂ）７１２２、およびＡＶ再生ライブラリ（ＡＶ再生Ｌｉｂ）７１２３を取り上げる。これらは、図４に示すリソース制御部４７の機能を実現するものである。

チューナＬｉｂ７１２１は、チューナ１０１を制御する機能をサブプログラムに提供する。すなわち、チューナＬｉｂ７１２１は、チューナに与える周波数を含むチューニング情報をサブプログラムから受け取る。そして、チューナＬｉｂ７１２１は、受け取った情報をチューナ１０１に与えるとともに、チューニングを行うようにチューナ１０１に指示する。

デスクランブラＬｉｂ７１２２は、デスクランブラ１０２を制御する機能をサブプログラムに提供する。すなわち、デスクランブラＬｉｂ７１２２は、鍵等を含む復号情報をサブプログラムから受け取る。そして、デスクランブラＬｉｂ７１２２は、受け取った情報をデスクランブラ１０２に与えるとともに、デスクランブラ１０２に復号を行うように指示する。

ＡＶ再生Ｌｉｂ７１２３は、ＴＳデコーダ１０３を制御

し、映像・音声再生を行う機能をサブプログラムに提供する。すなわち、ＡＶ再生Ｌｉｂ７１２３は、再生すべき映像のパケットＩＤおよび音声のパケットＩＤをサブプログラムから受け取る。そして、ＡＶ再生Ｌｉｂ７１２３は、受け取った映像のパケットＩＤと出力先「ビデオデコーダ１０６」との組、および、受け取った音声のパケットＩＤと出力先「オーディオデコーダ１０４」との組をＴＳデコーダ１０３に与えると共に、ＴＳデコーダ１０３にフィルタリングを行うように指示する。これによって、音声データを含むパケットがオーディオデコーダ１０４に引き渡された後、音声データはスピーカ１０５を通して再生される。また、映像データを含むパケットがビデオデコーダ１０６に引き渡された後、映像データは、ディプレイ１０７を通して再生される。

ナビゲータ７２０は、例えばユーザの指示に従って、フォーマットエンジンの起動や停止をフォーマットエンジンに対して指示する。また、ナビゲータ７２０は、デジタルテレビのチャンネル切り替えや、フォーマットエンジンでのアプリケーションの実行・停止、データの再生・停止等を指示する。ナビゲータ７２０は、ＯＳ７１０のカーネル７１１によって起動されるサブプログラムである。

次に、ナビゲータ７２０によるチャンネルの切り替えの動作を説明する。なお、入力部１１１は、図３で示されるフロントパネルであるとする。ユーザがＣ＋ボタン３０８あるいはＣ－ボタン３０９を押下すると、ナビゲータ７２０は、押下されたボタンの識別子を受け取り、再生してい

るチャンネルを切り替える。ここで、Ｃ＋ボタン３０８はチャンネル番号を番号が大きくなる方へ変化させ、Ｃ－ボタン３０９はチャンネル番号を番号が小さくなる方へ変化させる。再生可能なチャンネルの番号が例えば「１」、「２」「４」「６」「８」で、現在チャンネル「６」を再生しているとする。ここで、ユーザがＣ＋ボタン３０８を押下するとナビゲータ７２０は、再生するチャンネルを「６」から「８」に変更する。また、ユーザがＣ－ボタン３０９を押下するとナビゲータ７２０は、再生するチャンネルを「６」から「４」に変更する。チャンネル変更は、チャンネルに対応する映像・音声を含むパケットＩＤをＯＳ７１０のライブラリ７１２に含まれるＡＶ再生Ｌｉｂ７１２３に与えることによって行われる。この動作は、現在市販されているテレビの基本的な機能であるので、詳細な説明は省略する。

次に、図８～図１６を用いて、フォーマットエンジン等の起動指示がユーザによって行われる際のナビゲータの動作を説明する。ユーザが入力部１１１のＭＥＮＵボタン３０７を押下すると、ナビゲータ７２０は、実行可能なフォーマットエンジンの一覧をディスプレイ１０７に表示する。図８、図９および図１０は、ディスプレイ１０７に表示されるフォーマットエンジンの一覧の例を示す図である。図８～図１０では、フォーマットエンジンの一覧として、フォーマットエンジン名とその状態との組が表示される。さらに、フォーマットエンジンを選択するためにカーソル８１が表示される。カーソル８１は、ユーザが入力部１１

１の上カーソルボタン３０１または下カーソルボタン３０２を押下することに応じて移動する。図８に示す状態で下カーソルボタン３０２が押下されると、カーソル８１は１つ下に移動して図９に示す状態になる。図９の状態で下カーソルボタン３０２が押下されると、カーソル８１は１つ下に移動して図１０に示す状態になる。図９に示す状態で上カーソルボタン３０１が押下されると、カーソル８１は１つ上に移動して図８に示す状態になる。なお、図８～図１０において、表示される各フォーマットエンジン（Ｊａｖａミドルウェア７４０、ＨＴＭＬブラウザ７５０、およびメーラー７６０）の状態は、ナビゲータ７２０によって結合部７３０から取得される。

また、図８に示す状態でユーザが入力部１１１のＯＫボタン３０５を押下すると、ナビゲータ７２０は、Ｊａｖａミドルウェア７４０が実行可能なアプリケーションの一覧を表示する。図１１および図１２は、ディスプレイ１０７に表示される、実行可能なアプリケーションの一覧の例を示す図である。図１１および図１２では、実行可能なアプリケーションの一覧として、アプリケーション名とその状態との組が表示される。さらに、アプリケーションを選択するためにカーソル８１が表示される。カーソル８１は、ユーザが入力部１１１の上カーソルボタン３０１または下カーソルボタン３０２を押下することに応じて移動する。図１１の状態で下カーソルボタン３０２が押下されると、カーソル８１は１つ下に移動して図１２の状態になる。図１２の状態で上カーソルボタン３０１が押下されると、カ

ーソル８１は１つ上に移動して図１１の状態になる。なお、図１１および図１２において、表示される各アプリケーションの状態は、ナビゲータ７２０によって結合部７３０から取得される。

また、図１１の状態でユーザがＯＫボタン３０５を押下すると、ナビゲータ７２０は、アプリケーション「ＥＰＧ」を起動するように結合部７３０に対して指示する。図１２の状態でＯＫボタン３０５を押下すると、ナビゲータ７２０は、アプリケーション「トランプゲーム」を起動するように結合部７３０に対して指示する。

図１１および図１２では、全てのアプリケーションが停止中であったが、アプリケーションが「一時停止中」あるいは、「実行中」の場合もある。図１３および図１４は、ディスプレイ１０７に表示される、実行可能なアプリケーションの一覧の他の例を示す図である。図１３および図１４では、「ＥＰＧ」の動作状態が「実行中」であり、「トランプゲーム」の動作状態が「一時停止中」である。図１３の状態でユーザが入力部１１１のＯＫボタン３０５を押下すると、ナビゲータ７２０は、結合部７３０にアプリケーション「ＥＰＧ」を停止するように指示する。図１４の状態でユーザが入力部１１１のＯＫボタン３０５を押下すると、ナビゲータ７２０は、結合部７３０にアプリケーション「トランプゲーム」を停止するように指示する。まとめると、「停止中」であるアプリケーションをカーソル１１０によって選択している状態で、ユーザが入力部１１１のＯＫボタン３０５を押下すると、ナビゲータ７２０は、

当該アプリケーションを起動するように結合部７３０に対して指示する。一方、「実行中」あるいは「一時停止中」であるアプリケーションをカーソル８１によって選択している状態で、ユーザが入力部１１１のＯＫボタン３０５を押下すると、ナビゲータ７２０は、結合部７３０にそのアプリケーションを停止するように指示する。

また、図９の状態で、ユーザが入力部１１１のＯＫボタン３０５を押下すると、ナビゲータ７２０は、ＨＴＭＬブラウザ７５０が表示可能なデータの一覧を表示する。図１５および図１６では、実行可能なデータの一覧として、データ名とその状態との組が表示される。さらに、ユーザがデータを選択するためにカーソル８１が表示される。カーソル８１は、ユーザが入力部１１１の上カーソルボタン３０１または下カーソルボタン３０２を押下することにより移動する。図１５の状態で下カーソルボタン３０２が押下されると、カーソル８１は１つ下に移動して図１６の状態になる。図１６の状態で上カーソルボタン３０１が押下されると、カーソル８１は１つ上に移動して図１５の状態になる。なお、表示される各データの状態は、ナビゲータ７２０によって結合部７３０から取得される。また、図１５の状態でユーザがＯＫボタン３０５を押下すると、ナビゲータ７２０は、結合部７３０にデータ「天気予報」を停止するように指示する。図１６の状態でユーザがＯＫボタン３０５を押下すると、ナビゲータ７２０は、結合部７３０にデータ「レジャー情報」を停止するように指示する。

また、図１０の状態でユーザが入力部１１１のＯＫボタ

ン３０５を押下すると、ナビゲータ７２０は、メーラー７６０を起動するように結合部７３０に対して指示する。メーラー７６０は、取り扱う個々のデータに関する動作状態（例えば、停止中、実行中、一時停止中）を持っておらず、メーラー７６０自身の動作状態のみが規定されている。従って、図１０の状態でユーザが入力部１１１のＯＫボタン３０５を押下しても、Ｊａｖａミドルウェア７４０やＨＴＭＬブラウザ７５０のように個々のフォーマットエンジンが取り扱うアプリケーションやデータの一覧を表示することはない。ナビゲータ７２０は、メーラー７６０が取り扱うアプリケーションやデータの状態の一覧を結合部７３０から取得する際、メーラー７６０が個々のアプリケーションやデータの動作状態を保持していないことを知る。あるいは、ナビゲータ７２０は、メーラー７６０が個々のアプリケーションやデータの動作状態を保持していないことを予め知っていてもよい。

以上のように、ナビゲータ７２０は、個々のアプリケーションやデータの動作状態を保持していないフォーマットエンジンに対しては、ナビゲータ７２０は、次のように動作する。すなわち、フォーマットエンジンの動作状態が「停止中」であるフォーマットエンジンがカーソル８１によって選択されている状態でユーザが入力部１１１のＯＫボタン３０５を押下すると、ナビゲータ７２０は、当該フォーマットエンジンを起動するように結合部７３０に対して指示する。一方、「実行中」あるいは「一時停止中」であるフォーマットエンジンがカーソル８１によって選択され

ている状態でユーザが入力部１１１のＯＫボタン３０５を押下すると、ナビゲータ７２０は、当該フォーマットエンジンを停止するように結合部７３０に対して指示する。

なお、フォーマットエンジンあるいはフォーマットエンジンが取り扱うアプリケーションやデータの一覧が表示されている場合（図８～図１６）において、ユーザが入力部１１１のＭＥＮＵボタン３０７を押下すると、ナビゲータ７２０は、当該一覧の表示を取りやめる。

図４の説明に戻り、結合部７３０は、ナビゲータ７２０と複数のフォーマットエンジンとを結合するためのモジュールである。結合部７３０は、通信部７３１と、状態管理部７３２と、リソース管理部７３３とを備えている。

通信部７３１は、ナビゲータ７２０と各フォーマットエンジンとの間の通信を行う。ここで、通信とはメッセージデータの送受信を意味する。具体的には、ナビゲータ７２０がフォーマットエンジンに対して指示を行う場合、ナビゲータ７２０は当該指示を示すメッセージを通信部７３１を介して当該フォーマットエンジンへ送る。例えば、ナビゲータ７２０が、Ｊａｖａミドルウェア７４０が実行しているアプリケーションを停止する旨の指示を行う場合、ナビゲータ７２０は、当該停止の指示を意味するメッセージを通信部７３１を通してＪａｖａミドルウェア７４０に送る。

なお、ナビゲータ７２０や各フォーマットエンジンは、異なるプロセスあるいはスレッドで動作しているので、プロセス間通信あるいはスレッド間通信を行う必要がある。

プロセス間通信あるいはスレッド間通信は、ＯＳがＬｉｎｕｘであれば、Ｓｏｃｋｅｔ、ＰＩＰＥ等のＯＳ７１０が備える機能を利用することによって実現することができる。

ここで、もし、ナビゲータ７２０と各フォーマットエンジンとの間で個別に通信方法を決定することとすると、ナビゲータ７２０および各フォーマットエンジンは、複数種類の通信方法を実装しなければならない。従って、全プログラムのコード量が増え、開発工数も大きくなる。そこで、通信部７３１は、ナビゲータ７２０および各フォーマットエンジンが使用する共通の通信方法を規定する。ここでは、ナビゲータ７２０はＳｏｃｋｅｔ通信を用いることとするが、ＦＩＦＯ等、他の方法を用いても実現可能である。通信部７３１は、さらにＳｏｃｋｅｔ通信のプロトコル上に、独自のメッセージフォーマットを規定する。

次に、図１７～図３１を用いて、メッセージフォーマットについて詳細を説明する。図１７は、メッセージフォーマットの一例を示す図である。図１７において、メッセージは５つのフィールドで構成される。Ｓｏｕｒｃｅ　ＩＤフィールド１７１は、メッセージの送信元であるサブプログラムを示す情報を格納する。なお、サブプログラムには、サブプログラムを識別するためのサブプログラムＩＤが割り当てられる（後述する図１８参照。）。Ｓｏｕｒｃｅ　ＩＤフィールド１７１は、メッセージの送信元であるサブプログラムのサブプログラムＩＤを格納する。具体的には、Ｓｏｕｒｃｅ　ＩＤフィールド１７１は、メッセージ

を発信したナビゲータ７２０やフォーマットエンジンを示すサブプログラムＩＤを１バイトで格納する。サブプログラムＩＤの詳細な内容は後述する。

Ｄｅｓｔｉｎａｔｉｏｎ　ＩＤフィールド１７２は、メッセージを受け取るサブプログラムを示すサブプログラムＩＤ（ＳＩＤ）を１バイトで格納する。フォーマットエンジンＩＤの詳細な内容は後述する。Ｍｅｓｓａｇｅ　ＩＤフィールド１７３は、メッセージの内容を表すメッセージＩＤを１バイトで格納する。メッセージＩＤの詳細な内容は後述する。Ｄａｔａ　Ｌｅｎｇｈフィールド１７４は、Ｄａｔａフィールド１７５に格納されるデータの長さを２バイトで格納する。Ｄａｔａフィールド１７５は、メッセージＩＤに対応する詳細データを格納する。格納するデータのフォーマットはメッセージＩＤに対応して規定される。

図１８は、図１７に示すＳｏｕｒｃｅ　ＩＤフィールド１７１およびＤｅｓｔｉｎａｔｉｏｎ　ＩＤフィールド１７２に格納されるサブプログラムＩＤの表の一例を示す図である。図１８に示す表では、左側の列にサブプログラムが記載され、右側の列に対応するサブプログラムＩＤが記載されている。なお、左側の列にある「ＡＬＬ」は、全てのフォーマットエンジンを示す。ここでは、「ＡＬＬ」の他、ナビゲータ７２０、結合部７３０、Ｊａｖａミドルウェア７４０、ＨＴＭＬブラウザ７５０、およびメーラー７６０に対してサブプログラムＩＤが付されている。つまり、これらのサブプログラムがメッセージのやり取りを行う

サブプログラムである。

図１９は、Ｍｅｓｓａｇｅ　ＩＤフィールド１７３に格納されるメッセージＩＤの表の一例を示す図である。図１９においては、左側の列にメッセージ内容が記載され、中央の列にメッセージＩＤ（ＭＩＤ）が記載され、右側の列にＤａｔａフィールド１７５に格納するデータのフォーマット-ＩＤが記載されている。以下、各メッセージの詳細について説明する。

図１９において、上から２番目の行は、フォーマットエンジンに対して、フォーマットエンジンの状態を通知するように要求するための「フォーマットエンジン状態要求」メッセージが記載されている。例えば、ナビゲータ７２０が各フォーマットエンジンの動作状態を知りたいときに使用する。このメッセージはデータフィールド１７５を使用しない。

図１９において、上から３番目の行は、「フォーマットエンジン状態要求」の応答を行うための「フォーマットエンジン状態応答」メッセージである。例えば、Ｊａｖａミドルウェア７４０がナビゲータ７２０から「フォーマットエンジン状態要求」を受け取ると、Ｊａｖａミドルウェア７４０は、この「フォーマットエンジン状態応答」メッセージを用いてフォーマットエンジンの状態を通知する。このメッセージは、データフィールド１７５にフォーマットエンジンの状態を格納する。図２０は、フォーマットエンジン状態応答メッセージにおけるデータフィールド１７５のフォーマットの一例を示す図である。データフィールド

１７５には、フォーマットエンジンの動作状態を示す動作状態ＩＤが１バイトで格納される。

図２１は、フォーマットエンジンの動作状態と動作状態ＩＤとの対応を示す表の一例を示す図である。図２１において、左側の列にフォーマットエンジンの動作状態が記載され、中央の列に動作状態の意味が記載され、右側の列に動作状態ＩＤが記載されている。ここでは、「実行中」、「一時停止中」、および「停止中」という３つの動作状態が規定されている。図２１において規定されいるこれらの動作状態は、全フォーマットエンジンに共通する共通状態である。つまり、メッセージに含まれる動作状態は、共通状態の表現で表されている。また、「実行中」という共通状態は、デジタルテレビ１００が有している極小リソースを使用していることを意味する。

図１９の説明に戻り、上から４番目の行は、フォーマットエンジンの状態が変化したことを、フォーマットエンジンがナビゲータ７２０や他のフォーマットエンジンに通知するための「フォーマットエンジン状態変化」メッセージである。このメッセージはデータフィールド１７５を使用しない。

また、上から５番目から７番目の行は、ナビゲータ７２０等が他のフォーマットエンジンに対して状態を「実行中」「停止中」「一時停止中」に変更するように指示を出すための「フォーマットエンジン実行」「フォーマットエンジン停止」「フォーマットエンジン一時停止」メッセージである。これらのメッセージはデータフィールド１７５を

使用しない。

また、上から８番目の行は、フォーマットエンジンに対して、フォーマットエンジンが実行可能なアプリケーションあるいは表示可能なデータの一覧を通知するように要求するための「アプリケーション・データ一覧要求」メッセージである。例えば、Ｊａｖａミドルウェア７４０が実行可能なアプリケーションの一覧を、ナビゲータ７２０が知りたい場合に使用される。このメッセージはデータフィールド１７５を使用しない。

また、上から９番目の行は、「アプリケーション・データ一覧要求」の応答を行うための「アプリケーション・データ一覧応答」メッセージである。例えば、Ｊａｖａミドルウェア７４０がナビゲータ７２０から「アプリケーション・データ一覧要求」を受け取ると、Ｊａｖａミドルウェア７４０は、「アプリケーション・データ一覧応答」メッセージを用いて、Ｊａｖａミドルウェア７４０が実行可能なアプリケーションとその状態との一覧を通知する。このメッセージはデータフィールド１７５にアプリケーション・データの一覧を格納する。

図２２は、アプリケーション・データ一覧応答メッセージにおけるデータフィールド１７５のフォーマットの一例を示す図である。図２２においては、アプリケーション数（図２２では「アプリ数」と記載している）フィールド２０１に、フォーマットエンジンが実行可能なアプリケーションあるいは表示可能なデータの数が１バイトで格納される。

データフィールド１７５には、アプリケーション数フィールド２２０１に続いて、アプリケーション数フィールド２２０１に格納された数のアプリケーション情報フィールド２２０２が設けられる。

アプリケーション情報フィールド２２０２の先頭には、アプリケーションＩＤフィールド２２０３が設けられる。アプリケーションＩＤフィールド２２０３には、実行可能なアプリケーションあるいは表示可能なデータを識別するアプリケーションＩＤが１バイトで格納される。このアプリケーションＩＤは、フォーマットエンジンが割り当てる。

続いて、アプリケーション情報フィールド２２０２には、アプリケーション状態（図２２では「アプリ状態」と記載している）ＩＤフィールド２２０４が設けられる。アプリケーション状態ＩＤフィールド２２０４には、アプリケーションあるいはデータの状態を表すアプリケーション状態ＩＤを１バイトで格納する。なお、本構成例では、アプリケーションの状態とアプリケーション状態ＩＤとの対応は、図２１に示す対応、すなわち、フォーマットエンジンの動作状態と動作状態ＩＤとの対応と同じであるとする。

続いて、アプリケーション情報フィールド２２０２には、アプリケーション名長さ（図２２では「アプリ名の長さ」と記載している）フィールド２２０５が設けられる。アプリケーション名長さフィールド２２０５には、アプリケーションあるいはデータの名前の長さが格納される。

アプリケーション情報フィールド２２０２の末尾には、

アプリケーション名（図２２では「アプリ名」と記載している）フィールド２２０６が設けられる。アプリケーション名フィールド２２０６には、アプリケーション名長さフィールド２２０５に格納されたデータにより指定される長さのアプリケーションの名前が格納される。

図１９の説明に戻り、上から１０番目の行は、フォーマットエンジンが実行中のアプリケーションあるいは表示中のデータの状態が変化したことを、フォーマットエンジンがナビゲータ７２０や他のフォーマットエンジンに通知するための「アプリケーション・データ一覧変化」メッセージである。このメッセージはデータフィールド１７５を使用しない。

また、上から１１番目から１３番目の行は、ナビゲータ７２０等が他のフォーマットエンジンが実行可能なアプリケーションあるいは表示可能なデータに対して状態を「実行中」「停止中」「一時停止中」に変更するように指示を出すための「アプリケーション・データ実行」「アプリケーション・データ停止」「アプリケーション・データ一時停止」メッセージである。これらのメッセージはデータフィールド１７５に状態を変化させる対象のアプリケーションあるいはデータを識別するためのアプリケーションＩＤを格納する。

図２３は、「アプリケーション・データ実行」「アプリケーション・データ停止」「アプリケーション・データ一時停止」のメッセージにおけるデータフィールド１７５のフォーマットの一例を示す図である。データフィールド１

７５には、アプリケーションあるいはデータを識別するためのアプリケーションＩＤ（図２３では、「アプリＩＤ」と記載している）が１バイトで格納される。アプリケーションＩＤは、「アプリケーション・データ一覧応答」メッセージでフォーマットエンジンが返す値を用いる。

図１９の説明に戻り、上から１４番目の行は、ナビゲータ７２０や他のフォーマットエンジン間で、独自データの送受信時に使用する「プライベート」メッセージである。データフィールド１７５は、ナビゲータ７２０や他のフォーマットエンジン間で独自に規定したフォーマットに独自のデータを埋め込んで使用される。

なお、本構成例では、フォーマットエンジンの動作状態の規定と、アプリケーション・データの動作状態の規定とが同じであるが、異なっていても実施可能である。また、本構成例で規定しているメッセージおよびメッセージのフォーマットは一例であり、これ以外のメッセージを含む、他のフォーマットを使用する、あるいは本構成例で示したメッセージの一部・全部を使用しない等してもよい。重要なポイントは、結合部７３０あるいは同等の構成要素が、ナビゲータ７２０や複数のフォーマットエンジン間で使用される、フォーマットエンジンの共通状態およびアプリケーション・データの共通状態を規定し、共通の通信手段を提供することにある。

次に、ナビゲータ７２０の動作に対応するメッセージの具体例を説明する。まず、ナビゲータ７２０が図８に示すようなフォーマットエンジンの一覧を表示する際には、ナ

ビゲータ７２０は、図２４～図２６に示す３つの「フォーマットエンジン状態要求」メッセージを結合部７３０に渡す。図２４～図２６は、「フォーマットエンジン状態要求」メッセージの具体例を示す図である。図２４に示すメッセージ２４００は、ナビゲータ７２０からＪａｖａミドルウェア７４０への「フォーマットエンジン状態要求」である。Ｓｏｕｒｃｅ　ＩＤフィールド２４０１にはナビゲータ７２０を表すサブプログラムＩＤとして「１」が格納される（図１８参照）。Ｄｅｓｔｉｎａｔｉｏｎ　ＩＤフィールド２４０２には、Ｊａｖａミドルウェア７４０を表すサブプログラムＩＤとして「２」が格納される（図１８参照）。Ｍｅｓｓａｇｅ　ＩＤフィールド２４０３には、「フォーマットエンジン状態要求」を表す「１」が格納される（図１９参照）。Ｄａｔａ　Ｌｅｎｇｔｈフィールド１５０４には、データの長さとして「０」が格納される。図２５に示すメッセージ２５００は、ナビゲータ７２０からＨＴＭＬブラウザ７５０への「フォーマットエンジン状態要求」である。メッセージ２４００とメッセージ２５００の違いは、Ｄｅｓｔｉｎａｔｉｏｎ　ＩＤフィールド１５１２に、ＨＴＭＬブラウザ７５０を表すＩＤ「３」を格納している（図１８参照）部分だけであり、他は同じである。図２６に示すメッセージ２６００は、ナビゲータ７２０からメーラー７６０への「フォーマットエンジン状態要求」である。メッセージ２４００とメッセージ２６００の違いは、Ｄｅｓｔｉｎａｔｉｏｎ　ＩＤ１５２２に、メーラー７６０を表すＩＤ「４」を格納している（図１８参照）

部分だけであり、他は同じである。

「フォーマットエンジン状態要求」メッセージが送信された後、ナビゲータ７２０は、結合部７３０から図２７～図２９に示す３つの「フォーマットエンジン取得応答」メッセージを受け取る。図２７～図２９は、「フォーマットエンジン取得応答」メッセージの具体例を示す図である。メッセージ２７００は、Ｊａｖａミドルウェア７４０からナビゲータ７２０への「フォーマットエンジン取得応答」である。Ｓｏｕｒｃｅ　ＩＤフィールド２７０１には、Ｊａｖａミドルウェア７４０を表すサブプログラムＩＤ「２」が格納される（図１８参照）。Ｄｅｓｔｉｎａｔｉｏｎ　ＩＤフィールド２７０２には、ナビゲータ７２０を表すサブプログラムＩＤとして「１」が格納される（図１８参照）。Ｍｅｓｓａｇｅ　ＩＤフィールド２７０３には、「フォーマットエンジン取得応答」を表すメッセージＩＤとして「２」が格納される（図１９参照）。Ｄａｔａ　Ｌｅｎｇｔｈフィールド２７０４にはデータの長さ「１」が格納される。データフィールド２７０５には、「停止中」を表す動作状態ＩＤとして「３」が格納される（図２１参照）。メッセージ２８００は、ＨＴＭＬブラウザ７５０からナビゲータ７２０への「フォーマットエンジン取得応答」である。メッセージ２７００とメッセージ２８００の違いは、Ｓｏｕｒｃｅ　ＩＤフィールド２８０１に、ＨＴＭＬブラウザ７５０を表すサブプログラムＩＤとして「３」が格納されている部分だけであり、他は同じである。メッセージ２７２０は、メーラー７６０からナビゲータ７２０へ

の「フォーマットエンジン取得応答」である。メッセージ２７００とメッセージ２９００の違いは、Ｓｏｕｒｃｅ ＩＤフィールド２９０１に、図１８を参照してメーラー７６０を表すサブプログラムＩＤとして「４」が格納されている部分だけであり、他は同じである。このように３つの「フォーマットエンジン状態要求」メッセージを結合部７３０に送り、３つの「フォーマットエンジン取得応答」メッセージを結合部７３０から受け取ることによって、ナビゲータ７２０は、図８に示すフォーマットエンジン一覧を表示することができる。

次に、ナビゲータ７２０が図１５に示すようなＨＴＭＬブラウザ７５０が表示可能なデータの一覧を表示する場合について説明する。この場合、ナビゲータ７２０は、図３０に示す「アプリケーション・データ一覧要求」メッセージを結合部７３０に渡す。図３０は、「アプリケーション・データ一覧要求」メッセージの具体例を示す図である。図３０に示すメッセージ３０００は、ナビゲータ７２０からＨＴＭＬブラウザ７５０への「アプリケーション・データ一覧要求」である。Ｓｏｕｒｃｅ ＩＤフィールド１７０１には、ナビゲータ７２０を表すサブプログラムＩＤとして「１」が格納される（図１８参照）。Ｄｅｓｔｉｎａｔｉｏｎ ＩＤフィールド１７０２には、ＨＴＭＬブラウザ７５０を表すサブプログラムＩＤとして「３」が格納される（図１８参照）。Ｍｅｓｓａｇｅ ＩＤフィールド１７０３には、「アプリケーション・データ一覧要求」を表すメッセージＩＤとして「１１」が格納される（図１９参

照）。Ｄａｔａ　Ｌｅｎｇｔｈフィールド１７０４にはデータの長さを示す「０」が格納される。

「アプリケーション・データ一覧要求」メッセージを送信した後、ナビゲータ７２０は、結合部７３０から図３１に示す「アプリケーション・データ取得応答」メッセージを受け取る。図３１は、「アプリケーション・データ取得応答」メッセージの具体例を示す図である。図３１に示すメッセージ３１００は、ＨＴＭＬブラウザ７５０からナビゲータ７２０への「アプリケーション・データ取得応答」である。Ｓｏｕｒｃｅ　ＩＤフィールド３１０１には、ＨＴＭＬブラウザ７５０を表すサブプログラムＩＤとして「３」が格納される（図１８参照）。Ｄｅｓｔｉｎａｔｉｏｎ　ＩＤフィールド３１０２には、ナビゲータ７２０を表すサブプログラムＩＤとして「１」が格納される（図１８参照）。Ｍｅｓｓａｇｅ　ＩＤ３フィールド１０３には、「アプリケーション・データ取得応答」を表すメッセージＩＤとして「１２」が格納される（図１９参照）。Ｄａｔａ　Ｌｅｎｇｔｈフィールド１６０４には、データの長さを示す「２７」が格納される。アプリケーション数フィールド３１０５には、ＨＴＭＬブラウザ７５０が表示可能なデータの数を示す「２」が格納される。

また、図３１において、アプリケーション数フィールド３１０５には、１つ目のデータに対するアプリケーション情報（図３１には、「アプリ情報」と記載している）フィールド３１１１と、２つ目のデータに対するアプリケーション情報（図３１には、「アプリ情報」と記載している）

フィールド３１１２とが含まれている。アプリケーションＩＤフィールド３１２１には、１つ目のデータを表すアプリケーションＩＤとして「１」が格納されている。アプリケーション名フィールド３１２４には、データの名前を示す「天気予報」が格納されている。ここで「天気予報」の各文字は２バイトコードで表現されており、結果、アプリケーション名長さは、４文字×２バイト＝８バイトとなっている。アプリケーションＩＤフィールド３１２５には２つ目のデータを表すアプリケーションＩＤとして「２」が格納されている。状態ＩＤフィールド３１２７には、「実行中」を表すアプリケーション状態ＩＤとして「３」が格納されている。アプリケーション名長さフィールド３１２８には、アプリケーション名の長さを示す「１２」が格納される。アプリケーション名フィールド３１２４には、データの名前を示す「レジャー情報」が格納されている。ここで「レジャー情報」の各文字は２バイトコードで表現されており、結果、アプリケーション名長さは、６文字×２バイト＝１２バイトとなっている。

以上で、メッセージフォーマットについての説明を終了する。

再び図４の説明に戻り、通信部７３１は、ナビゲータ７２０またはフォーマットエンジンからメッセージを受け取ると、メッセージに含まれているＤｅｓｔｉｎａｔｉｏｎＩＤフィールドを調べ、対応するフォーマットエンジンあるいはナビゲータ７２０に当該メッセージを配送する。例えば、Ｄｅｓｔｉｎａｔｉｏｎ　ＩＤフィールドに「０

」が格納されている場合、通信部７３１は、全てのフォーマットエンジンおよびナビゲータ７２０にメッセージを送信する（図１８参照）。従って、ナビゲータ７２０は、全てのフォーマットエンジンを終了させたい場合、Ｄｅｓｔｉｎａｔｉｏｎ　ＩＤフィールドが「０」であり、メッセージＩＤが「５」であるメッセージを通信部３１へ送ればよい。

状態管理部７３２は、フォーマットエンジンの動作状態、およびフォーマットエンジンが実行可能なアプリケーションおよび表示可能なデータの動作状態を保持する。具体的には、状態管理部７３２は、それらの情報を２次記憶部１０８あるいは１次記憶部１０９に記憶する。

図３２～図３４は、状態管理部７３２が保持している情報の一例を示す図である。図３２は、フォーマットエンジンに関する情報を表形式で表した例である。図３２において、最も左側の列は、各フォーマットエンジンを識別するためのフォーマットエンジンＩＤである。ここでは、各フォーマットエンジンのフォーマットエンジンＩＤは、図１８で規定されたサブプログラムＩＤと同じであるとする。左から２番目の列は、各フォーマットエンジンの名前である。左から３番目の列は、各フォーマットエンジンの動作状態を表す動作状態ＩＤである。図３２に示す表において、動作状態ＩＤは、図２１において規定される値によって表される。最も右側の列は、各フォーマットエンジンが実行可能なアプリケーションあるいは表示可能なデータに関する情報の格納位置を示すポインターである。実行可能な

アプリケーションあるいは表示可能なデータを持たないフォーマットエンジンに関しては、当該列は「ＮＵＬＬ」と記述される。

また、図３２に示す表において上から２番目の行には、Ｊａｖａミドルウェア７４０の情報が記述されている。すなわち、フォーマットエンジンＩＤが「２」、名前が「Ｊａｖａミドルウェア」、動作状態ＩＤが、「停止中」を意味する「３」、実行可能なアプリケーションに関する情報に対するポインターが１６進数表記で「０ｘ５６７８」と記述されている。上から３番目の行には、ＨＴＭＬブラウザ７５０の情報が記述されている。すなわち、フォーマットエンジンＩＤが「３」、名前が「ＨＴＭＬブラウザ」、動作状態ＩＤが、「停止中」を意味する「３」、表示可能なデータに関する情報に対するポインターが１６進数表記で「０ｘ７１６２」と記述されている。上から４番目の行には、メーラー７６０の情報が記述されている。すなわち、フォーマットエンジンＩＤが「４」、名前が「メーラー」、動作状態ＩＤが、「停止中」を意味する「３」と記述されている。また、アプリケーション・データに関する情報はないので、ポインターが「ＮＵＬＬ」と記述されている。

図３３は、Ｊａｖａミドルウェア７４０が実行可能なアプリケーションに関する情報を表形式で表した例を示す図である。この情報は、１次記憶部１０９中の格納位置０ｘ５６７８から格納されている。図３３において左側の列には、各アプリケーションに割り当てられたアプリケーショ

ンＩＤが記述される。中央の列には、アプリケーションの名前が記述される。右側の列には、アプリケーション状態ＩＤが記述される。上から２番目の行には、アプリケーションＩＤが「１」、名前が「ＥＰＧ」、アプリケーション状態ＩＤが、「停止中」を表す「３」と記述されている。上から３番目の行には、アプリケーションＩＤが「２」、名前が「トランプゲーム」、アプリケーション状態ＩＤが、「停止中」を表す「３」と記述されている。

図３４は、ＨＴＭＬブラウザ７５０が実行可能なデータに関する情報を表形式で表した例を示す図である。この情報は、１次記憶部１０９中の格納位置０ｘ７１６２から格納されている。図３４において左側の列には、各データに割り当てられたデータＩＤが記述されている。中央の列には、データの名前が記述されている。右側の列には、データの状態を表すアプリケーション状態ＩＤが記述されている。上から２番目の行には、アプリケーションＩＤが「１」、名前が「天気予報」、アプリケーション状態ＩＤが、「実行中」を表す「１」と記述されている。上から３番目の行には、アプリケーションＩＤが「２」、名前が「レジャー情報」、アプリケーション状態ＩＤが、「実行中」を表す「１」と記述されている。

再び図４の説明に戻る。上述したように、通信部７３１は、メッセージに含まれるＤｅｓｔｉｎａｔｉｏｎ　ＩＤフィールドに格納されたデータに従ってメッセージを配送する。ここで、「フォーマットエンジン状態要求」メッセージおよび「アプリケーション・データ一覧要求」メッセ

ージに関しては、通信部７３１は、Ｄｅｓｔｉｎａｔｉｏｎ　ＩＤフィールドを無視して、状態管理部７３２に配送してもよい。状態管理部７３２は、各フォーマットエンジンの状態を保持しているので、各フォーマットエンジンに代わって「フォーマットエンジン状態要求」メッセージ、「フォーマットエンジン状態応答」メッセージ、あるいは「アプリケーション・データ一覧応答」メッセージを生成することができる。状態管理部７３２がこれらのメッセージを生成することによって、プロセス間通信やスレッド間通信を省略することができるので、短時間での応答を実現することができる。

状態管理部７３２は、常に最新のフォーマットエンジンの動作状態を保持する。従って、状態管理部７３２は、必要に応じて各フォーマットエンジンから情報を取得する。すなわち、まず、デジタルテレビ１００に電源が投入されると、状態管理部７３２は、各フォーマットエンジンに対して「フォーマットエンジン状態要求」メッセージを送る。そして、状態管理部７３２は、各フォーマットエンジンから「フォーマットエンジン状態応答」メッセージを受け取り、受け取ったメッセージに基づいて各フォーマットエンジンの動作状態を１次記憶部１０９に記憶する。また、状態管理部７３２は、各フォーマットエンジンに対して「アプリケーション・データ一覧要求」メッセージを送る。そして、状態管理部７３２は、各フォーマットエンジンから「アプリケーション・データ一覧応答」メッセージを受け取り、受け取ったメッセージに基づいて各アプリケーシ

ョンおよび各データのアプリケーション状態を１次記憶部１０９に記憶する。

さらに、状態管理部７３２は、フォーマットエンジンから「フォーマットエンジン状態変化」メッセージを受け取ると、当該フォーマットエンジンに対して「フォーマットエンジン状態要求」メッセージを送る。そして、状態管理部7-３２は、メッセージを送ったフォーマットエンジンから「フォーマットエンジン状態応答」メッセージを受け取り、受け取ったメッセージに基づいて各フォーマットエンジンの動作状態を１次記憶部１０９に記憶する。

次に、図３５～図３８を用いて、メーラー７６０の動作状態が変化する場合におけるメッセージのやり取りを説明する。図３５は、メーラー７６０が送信する「フォーマットエンジン状態変化」メッセージの一例を示す図である。図３５に示すメッセージ３５００は、メーラー７６０から全フォーマットエンジンへの「フォーマットエンジン状態変化」メッセージの一例である。メッセージ３５００においては、Ｓｏｕｒｃｅ　ＩＤフィールド３５０１には、メーラー７６０を表すサブプログラムＩＤとして「４」が格納される（図１８参照）。Ｄｅｓｔｉｎａｔｉｏｎ　ＩＤフィールド３５０２には、全部を表すサブプログラムＩＤとして「０」が格納される（図１８参照）。Ｍｅｓｓａｇｅ　ＩＤフィールド３５０３には、動作状態ＩＤとして「フォーマットエンジン状態変化」を表す「３」が格納される（図１９参照）。Ｄａｔａ　Ｌｅｎｇｔｈフィールド３５０４には、データの長さを表す「０」が格納される。

状態管理部７３２は、図３５に示すメッセージ３５００を受け取ると、図３６に示されるメッセージ３６００をメーラー７６０に送る。図３６は、状態管理部７３２が送信するメッセージの一例を示す図である。図３６に示すメッセージ３６００においては、Ｓｏｕｒｃｅ　ＩＤフィールド３６０１には、結合部７３０を表すサブプログラムＩＤとして「５」が格納される（図１８参照）。Ｄｅｓｔｉｎａｔｉｏｎ　ＩＤフィールド３６０２には、メーラー７６０を表すサブプログラムＩＤとして「４」が格納される（図１８参照）。Ｍｅｓｓａｇｅ　ＩＤフィールド３６０３には、メッセージＩＤとして「フォーマットエンジン状態取得」を表す「１」が格納される（図１９参照）。Ｄａｔａ　Ｌｅｎｇｔｈフィールド３６０４には、データの長さを表す「０」が格納される。

メーラー７６０は、図３６に示すメッセージ３６００を受け取ると、図３７で示されるメッセージ３７００を結合部７３０の状態管理部７３２に送る。図３７は、メーラー７６０が送信するメッセージの一例を示す図である。図３７に示すメッセージ３７００においては、Ｓｏｕｒｃｅ　ＩＤフィールド３７０１には、メーラー７６０を表すサブプログラムＩＤとして「４」が格納される（図１８参照）。Ｄｅｓｔｉｎａｔｉｏｎ　ＩＤフィールド３７０２には、結合部７３０を表すサブプログラムＩＤとして「５」が格納される（図１８参照）。Ｍｅｓｓａｇｅ　ＩＤフィールド３７０３には、メッセージＩＤとして、「フォーマットエンジン取得応答」を表す「２」が格納される（図１９

参照）。Ｄａｔａ　Ｌｅｎｇｔｈフィールド３７０４には、データの長さを表す「１」が格納される。データフィールド３７０５には、動作状態ＩＤとして「実行中」を表す「１」が格納される（図２１参照）。

図３７に示すメッセージ３７００が結合部７３０の状態管理部７３２へ送信された結果、状態管理部７３２は、メーラー７６０の状態が「実行中」に変化したことを知ることができる。このとき、メッセージ３７００の受信前の状態が図３２に示す状態であったとすると、状態管理部７３２は、図３８に示すように、１次記憶部１０９に記憶している情報を更新する。図３８は、図３２の変化後の状態を示す図である。図３８では、メーラー７６０の動作状態ＩＤが「１」になっている。

また、状態管理部７３２は、フォーマットエンジンから「アプリケーション・データー覧変化」メッセージを受け取ると、「フォーマットエンジン状態変化」メッセージの場合と同様、最新の状態を取り直す。すなわち、状態管理部７３２は、メッセージを送ってきたフォーマットエンジンに対して、「アプリケーション・データー覧要求」メッセージを送る。そして、状態管理部７３２は、当該フォーマットエンジンから「アプリケーション・データー覧応答」メッセージを受け取り、受け取ったメッセージに基づいて各アプリケーションおよび各データのアプリケーション状態を１次記憶部１０９に記憶する。

以上のようにして、状態管理部７３２は、各フォーマットエンジン、フォーマットエンジンが実行するアプリケー

ション、およびフォーマットエンジンが表示するデータの最新の情報を保持することができる。

次に、図３９～図４４を用いて、リソース管理部７３３について説明する。なお、リソース管理部７３３は、上述した第２の動作例の動作を行う際に必要となる構成であって、第１の動作例の動作を行う場合には不要である。

リソース管理部７３３は、複数のフォーマットエンジン間で発生したリソースの競合を解決するための情報を提供する。図３９は、リソース管理部７３３の内部構成を示すブロック図である。図３９において、リソース管理部７３３は、プロセス記憶部３９０１と、優先度記憶部３９０２と、最新起動記憶部３９０３と、リソースＩＤ記憶部３９０４と、フォーマットエンジン特定部３９０５と、優先度情報提供部３９０６と、リソース剥奪通知部３９０７とを備えている。

プロセス記憶部３９０１は、２次記憶部１０８あるいは１次記憶部１０９によって実現され、各フォーマットエンジンを実行しているプロセスあるいはスレッドの情報を保持する。ここで、各フォーマットエンジンはプロセス上で実行されているとする。

図４０は、プロセス記憶部３９０１において記憶されている情報の一例を示す図である。図４０においては、プロセス記憶部３９０１は、各フォーマットエンジンを実行しているプロセスＩＤを表形式で記憶している。図４０において、左側の列には、フォーマットエンジンＩＤ（ＦＩＤ）が記述されている。右側の列には、各フォーマットエン

ジンを実行しているプロセスを識別するためのプロセスＩＤが記述されている。プロセスＩＤは、ＯＳ７１０のカーネル７１１によって生成されるプロセスのＩＤである。リソース管理部７３３は、カーネル７１１からプロセスＩＤを受け取ってプロセス記憶部３９０１に記憶する。一方、上から２番目の行には、Ｊａｖａミドルウェア７４０のフォーマットエンジンＩＤである「２」（図１８参照）と、Ｊａｖａミドルウェア７４０を実行しているプロセスのプロセスＩＤである「１００」とが記述されている。上から３番目の行には、ＨＴＭＬブラウザ７５０のフォーマットエンジンＩＤである「３」（図１８参照）と、ＨＴＭＬブラウザ７５０を実行しているプロセスのプロセスＩＤである「１１０」とが記述されている。上から４番目の行には、メーラー７６０のフォーマットエンジンＩＤである「４」（図１８参照）と、メーラー７６０を実行しているプロセスのプロセスＩＤである「１２０」とが記述されている。

図３９の説明に戻り、優先度記憶部３９０２は、２次記憶部１０８、１次記憶部１０９あるいはＲＯＭ１１０によって実現される。優先度記憶部３９０２は、各フォーマットエンジンの優先度を記憶する。

図４１は、優先度記憶部３９０２に記憶されている情報の一例を示す図である。図４１において、優先度記憶部３９０２は、各フォーマットエンジンの優先度を表形式で記憶している。図４１において、左側の列には、フォーマットエンジンＩＤ（ＦＩＤ）が記述されている。右側の列に

は、各フォーマットエンジンの優先度が記述されている。ここで、優先度は値が大きいほど優先度が高いとする。また、上から２番目の行には、Ｊａｖａミドルウェアのフォーマットエンジンｉｄである「２」と、それに対応する優先度である「２」とが記述されている。上から３番目の行には、ＨＴＭＬブラウザのフォーマットエンジンＩＤである「３」と、それに対応する優先度「１」とが記述されている。上から４番目の行には、メーラーのフォーマットエンジンＩＤである「４」と、それに対応する優先度「４」とが記述されている。

図３９の説明に戻り、最新起動記憶部３９０３は、２次記憶部１０８あるいは１次記憶部１０９によって実現される。最新起動記憶部３９０３は、ナビゲータ７２０から送られてきた「フォーマットエンジン実行」メッセージ、または「アプリケーション・データ実行」メッセージに含まれているＤｅｓｔｉｎａｔｉｏｎ　ＩＤフィールドに記述されたフォーマットエンジンＩＤを記憶する。つまり、最新起動記憶部３９０３は、最新起動情報を記憶する。最新起動情報とは、ユーザが最後に実行を要求したフォーマットエンジンを示す情報である。

図４２は、最新起動記憶部３９０３が記憶している情報の一例を示す図である。図４２では、最新起動記憶部３９０３は、メーラー７６０のフォーマットエンジンＩＤである「４」を格納している。図４２に示す状態において、ナビゲータ７２０が、結合部７３０を通してＨＴＭＬブラウザ７５０に「フォーマットエンジン実行」メッセージ、あ

るいはアプリケーション・データ実行」メッセージを送った場合を考える。この場合、リソース管理部７３３は、最新起動記憶部３９０３に記憶される情報を、ＨＴＭＬブラウザ７５０を表すフォーマットエンジンＩＤである「３」に更新する。

図３９の説明に戻り、リソースＩＤ記憶部３９０４は、２次記憶部１０８あるいはＲＯＭ１１０によって実現される。リソースＩＤ記憶部３９０４は、リソース管理部７３３が管理対象とするリソースを識別するためのリソースＩＤ（ＲＩＤ）を記憶する。

図４３は、リソースＩＤ記憶部３９０４が記憶している情報の一例を示す図である。図４３においては、リソースＩＤ記憶部３９０４は、リソースＩＤを表形式で記憶する。図４３において、左側の列にはリソースＩＤが記述されている。右側の列には、対応するリソースの名前が記述されている。図４３では、管理対象のリソースとして、チューナ１０１、ＴＳデコーダ１０３、オーディオデコーダ１０４、ビデオデコーダ１０６、ネットワークインターフェース１１２が規定されている。これらのリソースに対してリソースＩＤが割り当てられている。図４４は、リソースＩＤの定義をプログラミング言語であるＣ言語で記述した例を示す図である。

図３９の説明に戻り、フォーマットエンジン特定部３９０５は、リソースを要求しているフォーマットエンジンを特定する。各フォーマットエンジンは、ＯＳ７１０のライブラリ７１２が提供する機能を利用する。例えば、チュー

ナ１０１を制御するチューナＬｉｂ７１２１を呼び出し、特定の周波数帯にチューニングを行う。これを実現するために、フォーマットエンジンは、チューナＬｉｂ７１２１が用意する関数を呼び出す。フォーマットエンジンから要求があると、すなわち、フォーマットエンジンが関数の呼び出しを行うと、チューナＬｉｂ７１２１は、どのフォーマットエンジンが当該関数を呼び出しているのかをフォーマットエンジン特定部３９０５に問い合わせる。この問い合わせに応じて、フォーマットエンジン特定部３９０５は、関数を呼び出しているプロセスを特定する。そして、フォーマットエンジン特定部３９０５は、特定したプロセスのプロセスＩＤ、およびプロセス記憶部３９０１が記憶しているフォーマットエンジンＩＤとプロセスＩＤとの組を参照することによって、関数を呼び出したフォーマットエンジンを特定する。

優先度情報提供部３９０６は、優先度記憶部３９０２が記憶しているフォーマットエンジンの優先度と、最新起動記憶部３９０３が記憶している、ユーザが最後に起動したフォーマットエンジンの情報とをライブラリ７１２に提供する。ライブラリ７１２は、フォーマットエンジン特定部３９０５および優先度情報提供部３９０６を利用することによって、どのフォーマットエンジンにリソースを提供するかを決定することができる。以下、具体的に説明する。

例えば、チューナＬｉｂ７１２１は、Ｊａｖａミドルウェア７４０から呼び出され、チューニングを行っているとする。なお、チューナＬｉｂ７１２１は、フォーマットエ

ンジン特定部３９０５を利用することによって、Ｊａｖａミドルウェア７４０が現在チューナ１０１を利用していることを知っている。この状態において、メーラー７６０がチューナＬｉｂ７１２１を呼び出し、他の周波数帯にチューニングしようとする場合を考える。チューナＬｉｂ７１２１は、フォーマットエンジン特定部３９０５を利用することによって、メーラー７６０がチューナ１０１の機能を利用していることを知る。次に、チューナＬｉｂ７１２１は、優先度情報提供部３９０６から最後に起動したフォーマットエンジンの情報を取得する。そして、チューナＬｉｂ７１２１は、最後に起動したフォーマットエンジンがメーラー７６０であれば、メーラー７６０にチューナ１０１の利用を許可する。一方、最後に起動したフォーマットエンジンがＪａｖａミドルウェア７４０であれば、メーラー７６０に対してチューナ１０１の利用を拒否する。最後に起動したフォーマットエンジンがメーラー７６０でもＪａｖａミドルウェア７４０でもない場合は、優先度情報提供部３９０６からメーラー７６０およびＪａｖａミドルウェア７４０の優先度を取得する。メーラー７６０の優先度がＪａｖａミドルウェア７４０の優先度よりも高い場合、チューナＬｉｂ７１２１は、メーラー７６０にチューナ１０１の使用を許す。一方、メーラー７６０の優先度がＪａｖａミドルウェア７４０の優先度よりも低い場合、メーラー７６０に対して、チューナ１０１の使用を拒否する。以上のようにして、ライブラリ７１２は、どのフォーマットエンジンにリソースを提供するかを決定することができる。

なお、ライブラリ７１２が、最後に起動したフォーマットエンジンの情報や優先度情報をどのように用いるかは自由であり、ここに紹介したのはその一例である。どのフォーマットエンジンに対してリソースの利用を許可するかを決定するためのルールは、どのようなものであってもよい。例えば、ライブラリ７１２は、優先度情報のみを用いてリソースの利用を許可するフォーマットエンジンを決定してもよい。また、ライブラリ７１２は、先に使用しているフォーマットエンジンに対して利用を許可してもよい。逆に、ライブラリ７１２は、後から呼び出したフォーマットエンジンに対して利用を許可してもよい。

図３９において、リソース剥奪通知部３９０７は、リソースの利用が許可されなかったフォーマットエンジンに対して、リソースが剥奪されたことを通知する。例えば、Ｊａｖａミドルウェア７４０がチューナＬｉｂ７１２１を呼び出し、チューナ１０１を使用しているとき、メーラー７６０がチューナＬｉｂ７１２１を呼び出し、チューナ１０１の使用を試みた場合を考える。ここで、もし、チューナＬｉｂ７１２１がメーラー７６０にチューナ１０１の使用を許すと、Ｊａｖａミドルウェア７４０が希望している周波数帯とは異なる周波数帯にチューニングされる。従って、Ｊａｖａミドルウェア７４０上で実行されているアプリケーションは正しい動作を続けることができなくなるおそれがある。これを防ぐために、チューナＬｉｂ７１２１は、チューナリソースが剥奪されたことをＪａｖａミドルウェア７４０に通知するように、リソース剥奪通知部３９０

７に依頼する。この依頼の際には、リソースが剥奪されたことを通知すべきフォーマットエンジンのフォーマットエンジンＩＤとリソースＩＤとがリソース剥奪通知部３９０７に渡される。なお、ライブラリ７１２は、フォーマットエンジンＩＤをフォーマットエンジン特定部３９０５から取得することによって知ることができる。また、ライブラリ７１２は、リソースＩＤ記憶部３９０４を参照することによってライブラリ７１２を予め知っている。リソース剥奪通知部３９０７は、フォーマットエンジンＩＤおよびリソースＩＤを受け取ると、フォーマットエンジンＩＤにより示されるフォーマットエンジンにリソースが剥奪されたことを通知する。なお、通知を受けたフォーマットエンジンの動作は、各フォーマットエンジンに依存する。動作続行が困難なフォーマットエンジンであれば、実行中から停止中あるいは一時停止中に動作状態を変化させるようにしてもよい。以上でリソース管理部７３３の説明を終了する。

次に、図４５～図７１を用いて、各フォーマットエンジンの詳細を説明する。

まず、図４５～図５１を用いて、Ｊａｖａミドルウェア７４０について説明する。Ｊａｖａミドルウェア７４０は、Ｊａｖａアプリケーションを実行するフォーマットエンジンである。本構成例においては具体的な例として、ＤＶＢ－ＭＨＰ１．０規格（正式には、ＥＴＳＩ　ＴＳ　１０１　８１２　ＤＶＢ－ＭＨＰ仕様Ｖ１．０．２）を取り上げるが、例えばアプレットビューワ等、他の仕様に従って

動作するものでも実施可能である。本構成例では、Ｊａｖａミドルウェア７４０の詳細な動作にはＤＶＢ－ＭＨＰ１．０規格の内容が含まれる。従って、ここでは、概略のみを記述する。

図４５は、Ｊａｖａミドルウェア７４０の構成を示すブロック図である。Ｊａｖａミドルウェア７４０は、Ｊａｖａバーチャルマシン（ＶＭ）４５０１と、クラスライブラリ４５０２と、アプリケーションマネージャ４５０３とを備えている。

ＪａｖａＶＭ４５０１は、Ｊａｖａ（ＴＭ）言語で記述されたアプリケーションを逐次解析し実行するＪａｖａバーチャルマシンである。Ｊａｖａ言語で記述されたプログラムはバイトコードと呼ばれる、ハードウエアに依存しない中間コードにコンパイルされる。Ｊａｖａバーチャルマシンは、このバイトコードを実行するインタープリタである。また、一部のＪａｖａバーチャルマシンは、バイトコードをＣＰＵ１１３が理解可能な実行形式に翻訳してから、ＣＰＵ１１３に引き渡し、実行することも行う。クラスライブラリ４５０２およびアプリケーションマネージャ４５０３の一部または全部もＪａｖａ言語で記述されており、ＪａｖａＶＭ４５０１によって実行される。本構成例では、ＯＳ７１０のカーネル７１１は、最初に実行するＪａｖａアプリケーションとしてアプリケーションマネージャ４５０３を指定する。Ｊａｖａ言語の詳細は、書籍「Ｊａｖａ　Ｌａｎｇｕａｇｅ　Ｓｐｅｃｉｆｉｃａｔｉｏｎ（ＩＳＢＮ　０－２０１－６３４５１－１）」等の多くの書

籍で解説されている。従って、ここでは、その詳細を省略する。また、ＪａｖａＶＭ自体の詳細な動作などは、「Ｊａｖａ　Ｖｉｒｔｕａｌ　Ｍａｃｈｉｎｅ　Ｓｐｅｃｉｆｉｃａｔｉｏｎ（ＩＳＢＮ　０－２０１－６３４５１－Ｘ）」等の多くの書籍で解説されている。ここでは、その詳細を省略する。

クラスライブラリ４５０２は、ＲＯＭ１１０に格納されている複数のＪａｖａクラスライブラリの集合である。その多くは、ＯＳ７１０のライブラリ７１２が提供する機能に対応するＪａｖａアプリケーションが呼び出し可能なＪａｖａ　ＡＰＩを提供する。この結果、Ｊａｖａアプリケーションはデジタルテレビ１００が有する機能を利用することができる。例えば、ＤＶＢ－ＭＨＰ１．０規格では、チューナ１０１を使用するためのＡＰＩとしてｏｒｇ．ｄａｖｉｃ．ｍｐｅｇ．ｔｕｎｉｎｇパッケージを規定し、Ｊａｖａアプリケーションがチューナ１０１を制御できるようにしている。

アプリケーションマネージャ４５０３は、放送波中に多重化されているＪａｖａアプリケーションをダウンロードし、実行する。ＭＨＰ１．０規格によれば、ＡＩＴと呼ばれるＪａｖａアプリケーションを定義する表が放送波中に多重化されて伝送される。アプリケーションマネージャ４５０３は、このＡＩＴを最初にダウンロードする。ＡＩＴにはアプリケーションをダウンロードするための情報やアプリケーションの名前、アプリケーションの制御情報などが含まれる。

図４６は、ＡＩＴの主要部を表した模式図である。図４６において最も左側の列には、Ｊａｖａアプリケーション ＩＤが記述されている。左から２番目の列には、Ｊａｖａアプリケーションの制御情報が記述されている。制御情報には「ａｕｔｏｓｔａｒｔ」、「ｐｒｅｓｅｎｔ」、「ｋｉｌｌ」等がある。「ａｕｔｏｓｔａｒｔ」は、Ｊａｖａミドルウェア７４０がそのＪａｖａアプリケーションを即時にかつ自動的に実行することを意味する。「ｐｒｅｓｅｎｔ」は、自動実行しないことを意味する。「ｋｉｌｌ」は、Ｊａｖａアプリケーションを停止することを意味する。左から３番目の列は、Ｊａｖａアプリケーションのアプリケーション名が記述されている。最も右側の列には、Ｊａｖａアプリケーションの優先度が記述されている。Ｊａｖａアプリケーション間でリソースの競合が発生した場合、優先度の高いＪａｖａアプリケーションにリソースが優先的に割り当てられる。ここで、優先度の値が大きいほど優先度が高いこととする。

図４６において上から２番目および３番目の行には、Ｊａｖａアプリケーションの情報の組が記述される。上から２番目の行において定義されるＪａｖａアプリケーションについては、ＪａｖａアプリケーションＩＤが「１」であり、制御情報が「ａｕｔｏｓｔａｒｔ」であり、アプリケーション名が「ＥＰＧ」であり、優先度が「６４」である。上から３番目の行において定義されるＪａｖａアプリケーションについては、ＪａｖａアプリケーションＩＤが「２」であり、制御情報が「ｐｒｅｓｅｎｔ」であり、アプ

リケーション名が「トランプゲーム」であり、優先度「３２」である。

アプリケーションマネージャ４５０３は、ＡＩＴを受け取ると、アプリケーション名「ＥＰＧ」のＪａｖａアプリケーションをダウンロードし実行する。ここで、ＤＶＢ－ＭＨＰ１．０規格が規定するＪａｖａアプリケーションは、「Ｌｏａｄｅｄ」、「Ｐａｕｓｅｄ」、「Ａｃｔｉｖｅ」、および「Ｄｅｓｔｒｏｙｅｄ」という４つの動作状態を有している。

図４７は、Ｊａｖａアプリケーションの４つの動作状態および動作状態間の遷移を表す状態遷移図である。アプリケーションマネージャ４５０３がＪａｖａアプリケーションをダウンロードし、当該Ｊａｖａアプリケーションが１次記憶部１０９に格納されると、当該Ｊａｖａアプリケーションの動作状態は「Ｌｏａｄｅｄ」になる。この動作状態から他の動作状態への遷移は、ダウンロードしたＪａｖａアプリケーションが実装している「ｉｎｉｔＸｌｅｔ」、「ｓｔａｒｔＸｌｅｔ」、「ｐａｕｓｅＸｌｅｔ」、および「ｄｅｓｔｒｏｙＸｌｅｔ」というメソッドをアプリケーションマネージャ４５０３が呼び出すことにより実現する。

「Ｌｏａｄｅｄ」の動作状態でアプリケーションマネージャ４５０３が「ｉｎｉｔＸｌｅｔ」メソッドを呼び出すと、Ｊａｖａアプリケーションの動作状態は、「Ｐａｕｓｅｄ」に遷移する。「Ｐａｕｓｅｄ」の動作状態でアプリケーションマネージャ４５０３が「ｓｔａｒｔＸｌｅｔ」

メソッドを呼び出すと、Ｊａｖａアプリケーションの動作状態は「Ａｃｔｉｖｅ」に遷移する。「Ａｃｔｉｖｅ」の動作状態でアプリケーションマネージャ４５０３が「ｐａｕｓｅＸｌｅｔ」メソッドを呼び出すと、Ｊａｖａアプリケーションの動作状態は、「Ｐａｕｓｅｄ」に遷移する。どの動作状態の場合においても、アプリケーションマネージャ４５０３が「ｄｅｓｔｒｏｙＸｌｅｔ」メソッドを呼び出すと、Ｊａｖａアプリケーションの動作状態は「Ｄｅｓｔｒｏｙｅｄ」に遷移する。

アプリケーションマネージャ４５０３は、Ｊａｖａアプリケーションの状態遷移を、（１）放送波中のＡＩＴの制御情報、（２）Ｊａｖａアプリケーションの指示、および（３）ナビゲータ７２０が結合部７３０を通して送る指示、の３通りの指示に従って行う。

ＡＩＴの制御情報が「ａｕｔｏｓｔａｒｔ」の場合、アプリケーションマネージャ４５０３は、Ｊａｖａアプリケーションをダウンロードした後、「ｉｎｉｔＸｌｅｔ」メソッド、「ｓｔａｒｔＸｌｅｔ」メソッドを呼び出し、「Ａｃｔｉｖｅ」状態に遷移させる。ＡＩＴは、時間と共に変化する。例えば、ＡＩＴが図４６から図４８に変化したとする。図４８では、「ＥＰＧ」という名前のアプリケーションの制御情報が「ａｕｔｏｓｔａｒｔ」から「ｋｉｌｌ」に変更されている。このとき、アプリケーションマネージャ４５０３は、Ｊａｖａアプリケーションの「ｄｅｓｔｒｏｙＸｌｅｔ」メソッドを呼び出し、「Ｄｅｓｔｒｏｙｅｄ」状態に遷移させる。

一方、Ｊａｖａアプリケーションは、自分自身の状態や他のＪａｖａアプリケーションの状態を遷移させることができる。ＤＶＢ－ＭＨＰ１．０規格では、Ｊａｖａアプリケーションが自身の状態を遷移させたいときは、Ｊａｖａアプリケーションは、「ｎｏｔｉｆｙＤｅｓｔｒｏｙｅｄ」、「ｎｏｔｉｆｙＰａｕｓｅｄ」、および「ｒｅｓｕｍｅＲｅｑｕｅｓｔ」メソッドのいずれかを呼び出す。「Ｄｅｓｔｒｏｙｅｄ」状態に遷移するには、Ｊａｖａアプリケーションは、「ｎｏｔｉｆｙＤｅｓｔｒｏｙｅｄ」を呼び出す。「Ａｃｔｉｖｅ」状態に遷移させるには、Ｊａｖａアプリケーションは、「ｒｅｓｕｍｅＲｅｑｕｅｓｔ」メソッドを呼び出す。「Ｐａｕｓｅｄ」状態に遷移するには、Ｊａｖａアプリケーションは、「ｎｏｔｉｆｙＰａｕｓｅｄ」を呼び出す。他のＪａｖａアプリケーションの状態を遷移させる場合、Ｊａｖａアプリケーションは、ＤＶＢ－ＭＨＰ１．０規格が規定しているｏｒｇ．ｄｖｂ．ａｐｐｌｉｃａｔｉｏｎパッケージが規定するメソッドを利用する。すなわち、まず、他のＪａｖａアプリケーションをアプリケーションＩＤによって指定した後、Ｊａｖａアプリケーションは、ｏｒｇ．ｄｖｂ．ａｐｐｌｉｃａｔｉｏｎパッケージが規定するメソッドを呼び出し、指定したＪａｖａアプリケーションの状態を遷移させる。ここでは詳細は省略する。

最後にナビゲータ７２０からの指示によってＪａｖａアプリケーションの状態を遷移する方法を説明する。ナビゲータ７２０は、共通状態、すなわち、結合部７３０が規定

しているフォーマットエンジンの動作状態またはフォーマットエンジンが実行可能なアプリケーションの動作状態に基づいて指示を出す。この共通状態は、個別状態、すなわち、Ｊａｖａミドルウェア７４０のアプリケーションマネージャ４５０３が実行するＪａｖａアプリケーションの動作状態とは異なっている。この共通状態と個別状態との間のギャップを第１の変換部７４１が埋めるのである。第１の変換部７４１は、ナビゲータ７２０からの指示を、Ｊａｖａミドルウェア７４０が取り扱う動作状態に変換する。その結果、アプリケーションマネージャ４５０３は、ＡＩＴの制御情報やＪａｖａアプリケーションからの指示と同等の指示を受け、Ｊａｖａアプリケーションの動作状態を遷移させることができる。

第１の変換部７４１は、結合部７３０からのメッセージを変換してＪａｖａミドルウェア７４０に伝え、また、Ｊａｖａミドルウェア７４０の情報を変換して結合部７３０に伝えることにより、Ｊａｖａミドルウェア７４０の動作を結合部７３０が規定する状態に適合させる。ここまで、様々なメッセージを、結合部７３０を通してＪａｖａミドルウェア７４０に配送すると記述してきたが、正確には、これらのメッセージは、Ｊａｖａミドルウェア７４０ではなく第１の変換部７４１に送信される。第１の変換部７４１が結合部７３０から「フォーマットエンジン状態要求」メッセージを受け取った場合、第１の変換部７４１は、Ｊａｖａミドルウェア７４０が実行可能な全てのＪａｖａアプリケーションの状態に基づいてＪａｖａミドルウェア７

４０の状態を決定する。そして、第１の変換部７４１は、「フォーマットエンジン状態応答」メッセージを作成し、結合部７３０へ送る。

図４９は、Ｊａｖａミドルウェア７４０が実行可能な全てのＪａｖａアプリケーションの動作状態と、Ｊａｖａミドルウェア７４０の動作状態との対応を示す変換表の一例を示す図である。図４９において左側の列には、Ｊａｖａミドルウェア７４０が実行可能な全てのＪａｖａアプリケーションの動作状態が記述されている。右側の列には、対応するＪａｖａミドルウェア７４０の動作状態が記述されている。ここで、Ｊａｖａミドルウェア７４０の状態は、結合部７３０が規定した状態、すなわち共通状態である。具体的には、図２１で示した「実行中」、「一時停止中」、および「停止中」という３つの動作状態である。上から２番目から４番目の行には、これら３つの動作状態に対応する、Ｊａｖａミドルウェア７４０が実行可能な全てのＪａｖａアプリケーションの動作状態が規定されている。

まず、上から２番目の行には、全てのＪａｖａアプリケーションの中に１つでも「Ａｃｔｉｖｅ」という動作状態のＪａｖａアプリケーションがあることに対しては、Ｊａｖａミドルウェア７４０の動作状態として「実行中」が対応付けられている。一般に、「Ａｃｔｉｖｅ」状態のＪａｖａアプリケーションは極小リソースを使用している可能性が高いため、このような対応関係を規定したものである。

次に、上から３番目の行には、「Ａｃｔｉｖｅ」状態の

Ｊａｖａアプリケーションがなく、かつ、全Ｊａｖａアプリケーションの中で１つでも「Ｐａｕｓｅｄ」状態のＪａｖａアプリケーションがあることに対して、Ｊａｖａミドルウェア７４０の動作状態として「一時停止中」が対応付けられている。ＤＶＢ－ＭＨＰ１．０規格によれば、「Ｐａｕｓｅｄ」状態のＪａｖａアプリケーションは、必要最小限のリソース以外は解放すべきと記述されている。よって、このような対応関係を規定した。

次に、上から４番目の行には、上から２番目および３番目の行に示す場合以外の場合に対して、「停止中」が対応付けられている。上から２番目および３番目の行に示す場合以外の場合とは、具体的には、全てのＪａｖａアプリケーションが、「Ｌｏａｄｅｄ」あるいは「Ｄｅｓｔｒｏｙｅｄ」という動作状態、あるいは読み込みも完了していない動作状態にある場合である。この場合、Ｊａｖａアプリケーションのコードは１つも実行されていないので、リソースを一切使用していない「停止中」に対応付けるのが妥当である。

前述した通り、Ｊａｖａミドルウェア７４０のアプリケーションマネージャ４５０３は、ＡＩＴの制御情報の変化やＪａｖａアプリケーションの指示により、Ｊａｖａアプリケーションの状態を遷移させる。第１の変換部７４１は、状態変化の通知をアプリケーションマネージャ４５０３から受ける。その結果としてＪａｖａミドルウェアの共通状態が変化する場合には、第１の変換部７４１は、図４９の変換表を参照して、「フォーマットエンジン状態変化」

メッセージを生成して、結合部７３０に通知する。この場合、第１の変換部７４１は、１次記憶部１０９にＪａｖａミドルウェアの共通状態を保存しておく。そして、第１の変換部７４１は、Ｊａｖａアプリケーションの状態が変化した後のＪａｖａミドルウェアの共通状態と比較する。なお、第１の変換部７４１は、Ｊａｖａアプリケーションの状態の変化をアプリケーションマネージャ４５０３から通知されたとき、常に「フォーマットエンジン状態変化」メッセージを生成して、結合部７３０に通知してもよい。

第１の変換部７４１が結合部７３０から「フォーマットエンジン実行」メッセージを受け取った場合、第１の変換部７４１は、ＡＩＴで規定された所定のＪａｖａアプリケーションを「Ａｃｔｉｖｅ」状態に遷移させるようにアプリケーションマネージャ４５０３に指示する。ここで所定のＪａｖａアプリケーションとは、ＡＩＴに規定されている全てのＪａｖａアプリケーションであってもよいし、優先度が最も高いＪａｖａアプリケーションであってもよい。所定のＪａｖａアプリケーションを「Ａｃｔｉｖｅ」状態に遷移させるには、ＤＶＢ－ＭＨＰ１．０規格が規定しているｏｒｇ．ｄｖｂ．ａｐｐｌｉｃａｉｔｏｎパッケージの機能を利用することができる。状態を変化させるべきＪａｖａアプリケーションのＩＤを指定した後、遷移させる状態「Ａｃｔｉｖｅ」を指定することによって、対象のＪａｖａアプリケーションの状態を「Ａｃｔｉｖｅ」状態に遷移させることができる。なお、すでに「Ａｃｔｉｖｅ」状態のＪａｖａアプリケーションが存在する場合、第１

の変換部７４１は何もしないこととしてもよい。

第１の変換部７４１が結合部７３０から「フォーマットエンジン停止」メッセージを受け取った場合、第１の変換部７４１は、ＡＩＴに規定された全てのＪａｖａアプリケーションを「Ｄｅｓｔｒｏｙｅｄ」状態に遷移させるようにアプリケーションマネージャ４５０３に指示する。ただし、読み込んでいないＪａｖａアプリケーションに対しては何もしなくてもよい。全てのＪａｖａアプリケーションを「Ｄｅｓｔｒｏｙｅｄ」状態に遷移させるには、ＤＶＢ－ＭＨＰ１．０規格が規定しているｏｒｇ．ｄｖｂ．ａｐｐｌｉｃａｉｔｏｎパッケージの機能を利用することができる。ＪａｖａアプリケーションのＩＤを指定した後、遷移させる状態「Ｄｅｓｔｒｏｙｅｄ」を指定することによって、対象のＪａｖａアプリケーションの状態を「Ｄｅｓｔｒｏｙｅｄ」状態に遷移させることができる。この操作をＡＩＴが定義する全Ｊａｖａアプリケーションに対して行えばよい。

第１の変換部７４１が結合部７３０から「フォーマットエンジン一時停止」メッセージを受け取った場合、第１の変換部７４１は、ＡＩＴに規定されたＪａｖａアプリケーションの中で「Ａｃｔｉｖｅ」状態のＪａｖａアプリケーションを見つけ、そのＪａｖａアプリケーションの状態を「Ｐａｕｓｅｄ」状態に遷移させるようにアプリケーションマネージャ４５０３に指示する。この処理を実現するには、ＤＶＢ－ＭＨＰ１．０規格が規定しているｏｒｇ．ｄｖｂ．ａｐｐｌｉｃａｉｔｏｎパッケージの機能を利用す

ることができる。ＪａｖａアプリケーションのＩＤを指定した後、そのＪａｖａアプリケーションの状態を取得する。もし状態が「Ａｃｔｉｖｅ」であれば、遷移させる状態「Ｐａｕｓｅｄ」を指定することによって、対象のＪａｖａアプリケーションの状態を「Ｐａｕｓｅｄ」状態に遷移させることができる。この操作をＡＩＴが定義する全てのＪａｖａアプリケーションに対して行えばよい。

第１の変換部７４１が結合部７３０から「アプリケーション・データ一覧要求」メッセージを受け取った場合、第１の変換部７４１は、ＡＩＴが定義している全てのＪａｖａアプリケーションおよびその状態に基づいて、「アプリケーション・データ一覧応答」メッセージを作成し、結合部７３０に送る。「アプリケーション・データ一覧応答」メッセージに含めるアプリケーション名は、ＡＩＴが規定するアプリケーション名がそのまま使用される。「アプリケーション・データ一覧応答」メッセージに含める状態ＩＤは、各Ｊａｖａアプリケーションの状態を結合部７３０が規定する共通状態に変換表を用いて変換することによって作成される。

図５０は、各Ｊａｖａアプリケーションの動作状態を共通状態に変換するための変換表の一例を示す図である。図５０において左側の列には、個別状態、すなわち、ＤＶＢ－ＭＨＰ１．０規格が規定しているＪａｖａアプリケーションの動作状態が記述されている。右側の列には、対応する共通状態が記述されている。上から２番目の行に示されるように、ＤＶＢ－ＭＨＰ１．０規格が規定しているＪａ

ｖａアプリケーションの動作状態「Ａｃｔｉｖｅ」は、「実行中」という共通状態に対応付けられている。一般に、「Ａｃｔｉｖｅ」状態のＪａｖａアプリケーションは極小リソースを使用している可能性が高いので、このような対応関係を規定した。

また、図５０において上から３番目の行に示されるように、ＤＶＢ－ＭＨＰ１．０規格が規定しているＪａｖａアプリケーションの動作状態「Ｐａｕｓｅｄ」は、「一時停止中」という共通状態に対応付けられている。ＤＶＢ－ＭＨＰ１．０規格によれば、「Ｐａｕｓｅｄ」状態のＪａｖａアプリケーションは、必要最小限のリソース以外は解放すべきと記述されている。よって、このような対応関係を規定した。

また、上から４番目の行に示されるように、上から２番目および３番目の行以外の場合のＪａｖａアプリケーションの動作状態に対しては、「停止中」という共通状態が対応付けられている。上から２番目および３番目の行以外の場合とは、具体的には、Ｊａｖａアプリケーションの動作状態が、Ｊａｖａアプリケーションが「Ｌｏａｄｅｄ」もしくは「Ｄｅｓｔｒｏｙｅｄ」状態、または読み込みも完了していない状態にある場合である。この場合、Ｊａｖａアプリケーションのコードは１つも実行されていないので、リソースを一切使用していない「停止中」に対応付けるのが妥当である。

今、アプリケーション名が「ＥＰＧ」であるＪａｖａアプリケーションの動作状態が「Ａｃｔｉｖｅ」であり、ア

プリケーション名が「トランプゲーム」であるＪａｖａアプリケーションの動作状態が「Ｄｅｓｔｒｏｙｅｄ」であるように、ＡＩＴにおいて定義されている場合を考える。この場合、第１の変換部７４１は、「アプリケーション・データ一覧要求」メッセージを受け取ると、図５１に示す「アプリケーション・データ一覧応答」メッセージ５１００を生成し、結合部７３０に送信する。

図５１は、第１の変換部７４１が送信するメッセージの具体例を示す図である。図５１に示すメッセージ５１００は、Ｊａｖａミドルウェア７４０からナビゲータ７２０への「アプリケーション・データ取得応答」である。Ｓｏｕｒｃｅ　ＩＤフィールド５１０１には、Ｊａｖａミドルウェア７４０を表すサブプログラムＩＤとして「２」が格納される（図１８参照）。Ｄｅｓｔｉｎａｔｉｏｎ　ＩＤフィールド５１０２には、ナビゲータ７２０を表すサブプログラムＩＤとして「１」が格納される（図１８参照）。Ｍｅｓｓａｇｅ　ＩＤフィールド５１０３には、「アプリケーション・データ取得応答」を表すメッセージＩＤとして「１２」が格納される（図１９参照）。Ｄａｔａ　Ｌｅｎｇｔｈフィールド５１０４には、データの長さを示す「２７」が格納される。アプリケーション数フィールド５１０５には、Ｊａｖａミドルウェア７４０が受け取ったＡＩＴに規定されているＪａｖａアプリケーション数である「２」が格納される。

また、図５１において、アプリケーション数フィールド５１０５には、１つ目のＪａｖａアプリケーションに対す

るアプリケーション情報フィールド５１１１と、２つ目のＪａｖａアプリケーションに対するアプリケーション情報フィールド５１１２とが含まれている。アプリケーションＩＤフィールド５１２１には、１つ目のＪａｖａアプリケーションを表すアプリケーションＩＤとして「１」が格納されている。アプリケーション状態ＩＤフィールド５１２２には、「実行中」を表すアプリケーション状態ＩＤとして「３」が格納されている。この「実行中」は、Ｊａｖａアプリケーションの動作状態「Ａｃｔｉｖｅ」が変換された結果である。アプリケーション名長さフィールド５１２３には、アプリケーション名の長さを示す「６」が格納される。アプリケーション名フィールド５１２４には、ＡＩＴで定義されたアプリケーション名「ＥＰＧ」が格納されている。ここで「ＥＰＧ」の各文字は２バイトコードで表現されており、結果、アプリケーション名長さは、３文字×２バイト＝６バイトとなっている。アプリケーションＩＤフィールド５１２５には２つ目のＪａｖａアプリケーションを表すアプリケーションＩＤとして「２」が格納されている。状態ＩＤフィールド５１２６には、「停止中」を表すアプリケーション状態ＩＤとして「３」が格納されている。この「停止中」は、Ｊａｖａアプリケーションの動作状態「Ｄｅｓｔｒｏｙｅｄ」が変換された結果である。アプリケーション名長さフィールド５１２７には、アプリケーション名の長さを示す「１４」が格納される。アプリケーション名フィールド５１２８には、ＡＩＴで定義されたアプリケーション名「トランプゲーム」が格納されてい

る。ここで「トランプゲーム」の各文字は２バイトコードで表現されており、結果、アプリケーション名長さは、７文字×２バイト＝１４バイトとなっている。

前述した通り、Ｊａｖａミドルウェア７４０のアプリケーションマネージャ４５０３は、ＡＩＴの制御情報の変化やＪａｖａアプリケーションの指示によってＪａｖａアプリケーションの動作状態を遷移させる。第１の変換部７４１は、状態変化の通知をアプリケーションマネージャ４５０３より受ける。その時、第１の変換部７４１は、「アプリケーション・データ一覧変化」メッセージを生成して、結合部７３０に通知する。第１の変換部７４１が結合部７３０から「アプリケーション・データ実行」メッセージを受け取った場合、第１の変換部７４１は、メッセージのデータフィールド１７５で指定されるＪａｖａアプリケーションの状態を「Ａｃｔｉｖｅ」状態に遷移させるようにアプリケーションマネージャ４５０３に指示する。Ｊａｖａアプリケーションを「Ａｃｔｉｖｅ」状態に遷移させるには、ＤＶＢ－ＭＨＰ１．０規格が規定しているｏｒｇ．ｄｖｂ．ａｐｐｌｉｃａｉｔｏｎパッケージの機能を利用することができる。状態を変化させるＪａｖａアプリケーションのＩＤを指定した後、遷移させる状態「Ａｃｔｉｖｅ」を指定することによって、対象のＪａｖａアプリケーションの状態を「Ａｃｔｉｖｅ」状態に遷移させることができる。なお、すでにＪａｖａアプリケーションが「Ａｃｔｉｖｅ」状態の場合、第１の変換部７４１は、何もしないようにしてもよい。

第１の変換部７４１が結合部７３０から「アプリケーション・データ停止」メッセージを受け取った場合、第１の変換部７４１は、メッセージのデータフィールド１７５で指定されるＪａｖａアプリケーションの状態を「Ｄｅｓｔｒｏｙｅｄ」に遷移させるようにアプリケーションマネージャ４５０３に指示する。ただし、読み込んでいないＪａｖａアプリケーションに対しては何もしなくてもよい。Ｊａｖａアプリケーションを「Ｄｅｓｔｒｏｙｅｄ」状態に遷移させるには、ＤＶＢ－ＭＨＰ１．０規格が規定しているｏｒｇ．ｄｖｂ．ａｐｐｌｉｃａｉｔｏｎパッケージの機能を利用することができる。ＪａｖａアプリケーションのＩＤを指定した後、遷移させる状態「Ｄｅｓｔｒｏｙｅｄ」を指定することによって、対象のＪａｖａアプリケーションの状態を「Ｄｅｓｔｒｏｙｅｄ」状態に遷移させることができる。

第１の変換部７４１が結合部７３０から「アプリケーション・データ一時停止」メッセージを受け取った場合、第１の変換部７４１は、メッセージのデータフィールド１７５で指定されるＪａｖａアプリケーションの状態を「Ｐａｕｓｅｄ」に遷移させるようにアプリケーションマネージャ４５０３に指示する。この処理を実現するには、ＤＶＢ－ＭＨＰ１．０規格が規定しているｏｒｇ．ｄｖｂ．ａｐｐｌｉｃａｉｔｏｎパッケージの機能を利用することができる。ＪａｖａアプリケーションのＩＤを指定した後、遷移させる状態「Ｐａｕｓｅｄ」を指定することによって、対象のＪａｖａアプリケーションの状態を「Ｐａｕｓｅｄ

」状態に遷移させることができる。

さらに、第１の変換部７４１は、結合部７３０のリソース管理部７３３からリソース剥奪の通知を受けてもよい。第１の変換部７４１は、剥奪されたリソースに応じて、アプリケーションマネージャ４５０３にＪａｖａアプリケーションの状態を遷移させることとしてもよい。Ｊａｖａアプリケーションの実行に不可欠なリソースが剥奪された場合、第１の変換部７４１は、全てのＪａｖａアプリケーションの状態を「Ｄｅｓｔｒｏｙｅｄ」に遷移させる。それ以外のリソース、すなわち、Ｊａｖａアプリケーションの実行に必ずしも必要ではないリソースが剥奪された場合については、第１の変換部７４１は、何も行わなくともよい。

また、リソース剥奪通知は、Ｊａｖａミドルウェア７４０が直接受け取り、対応する処理を行ってもよい。以上で、Ｊａｖａミドルウェア７４０の説明を終了する。

次に、図５２～図７１を用いて、ＨＴＭＬブラウザ７５０について説明する。ＨＴＭＬブラウザ７５０は、ＨＴＭＬデータを解釈し表示を行うフォーマットエンジンである。本構成例においては具体的な例として、ＤＶＢ－ＭＨＰ１．１規格（正式には、ＥＴＳＩ　ＴＳ　１０１　８１２　ＤＶＢ－ＭＨＰ仕様Ｖ１．１）が規定するＤＶＢ－ＨＴＭＬを取り上げるが、例えば、ＸＨＴＭＬ１．０等の他の仕様に従って動作するものでも実施可能である。本構成例では、ＨＴＭＬブラウザ４５０の詳細な動作にはＤＶＢ－ＭＨＰ１．１規格の内容が含まれる。従って、ここでは、

概略のみを記述する。

図５２は、ＨＴＭＬブラウザ７５０の構成を示すブロック図である。ＨＴＭＬブラウザ７５０は、パーザー５２０１と、レイアウター５２０２と、描画部５２０３と、インタラクション部５２０４と、ＨＴＭＬマネージャ５２０５とを備えている。

パーザー５２０１は、ＨＴＭＬマネージャ５２０５がダウンロードしたＤＶＢ－ＨＴＭＬデータの解析を行いＤＯＭ（ドキュメントオブジェクトモデル）ツリーを構築する。図５３は、ＤＶＢ－ＨＴＭＬデータの一例を示す図である。ＤＶＢ－ＨＴＭＬデータに対して、パーザー５２０１は図５５に示すようなＤＯＭツリーを構築する。

レイアウター５２０２は、パーザー５２０１が構築したＤＯＭツリーの各構成要素に対して、ディスプレイ１０７上の表示位置を計算して決定する。図５４は、ＤＶＢ－ＨＴＭＬデータのレイアウトを決めるスタイルシートの一例を示す図である。図５４に示す各行は各々、図５３に示すＤＶＢ－ＨＴＭＬデータ中の各要素５３１１，５３１２，５３１３，および５３１４の表示位置を指定している。ここでは、スタイルシートの例としてＤＶＢ－ＭＨＰが規定するＣＳＳ（Ｗ３Ｃが規定するＣＳＳ２の拡張仕様）を取り上げているが、レイアウトを決めようとするＨＴＭＬデータのスタイルシートとしてレイアウターが解析可能なものであれば、ＣＳＳ２等の他の仕様に従うスタイルシートであっても実施可能である。レイアウター５２０２は、スタイルシートに基づいて、ＤＯＭツリーの各構成要素に対

してディスプレイ１０７上の表示位置を決定する。例えば、図５５に示す“Ｗｅａｔｈｅｒ　ｆｏｒｅｃａｓｔ”に対して（５０，５０）で示される表示位置が決定された場合、文字列“Ｗｅａｔｈｅｒ　ｆｏｒｅｃａｓｔ”は、その左上の座標がディスプレイの上から５０ｐｘ，左から５０ｐｘの位置となるように表示される。図５５に示す“Ｊａｐａｎ“に対して（２００，１００）で示される表示位置が決定された場合、文字列“Ｊａｐａｎ“は、その左上の座標が、ディスプレイの上から２００ｐｘ，左から１００ｐｘの位置となるように表示される。

なお、レイアウターがスタイルシートのパーザーを持ち、ＤＯＭツリー構築後においてそのパーザーがＨＴＭＬマネージャから指示を受けることによってスタイルシートを解析・計算し、計算した結果の表示位置をレイアウターに通知してもよい。また、ＨＴＭＬブラウザがデフォルトのスタイルシートを持ち、スタイルシートが存在しない場合は、レイアウターは、ＨＴＭＬブラウザが持つデフォルトのスタイルシートに従い、ＤＯＭツリーの各構成要素に対して、ディスプレイ１０７上の表示位置を計算して決定してもよい。

図５２の説明に戻り、描画部５２０３は、レイアウター５２０２が計算した表示位置に従って、ＤＯＭツリーの各構成要素を描画する。図５６は、図５５に示すＤＯＭツリーを描画した際のディスプレイ１０７を示す図である。

インタラクション部５２０４は、描画したＤＶＢ－ＨＴＭＬデータに含まれるリンクやボタン等に対してユーザが

入力した指示に対応する処理を行う。例えば、ＤＶＢ－ＨＴＭＬデータ内に他のＤＶＢ－ＨＴＭＬデータへのリンクが定義されているとする。ここで、ユーザが、リンクのクリックを入力部１１１から入力すると、インタラクション部５２０４は、リンクに定義されたＤＶＢ－ＨＴＭＬデータの情報をＨＴＭＬマネージャ５２０５に引き渡す。

H-ＴＭＬマネージャ５２０５は、引き渡されたＤＶＢ－ＨＴＭＬデータの情報に基づいて、ＤＶＢ－ＨＴＭＬデータをダウンロードする。そして、ＨＴＭＬマネージャ５２０５は、パーザー５２０１、レイアウター５２０２、および描画部５２０３を用いてＤＶＢ－ＨＴＭＬデータを解釈・表示する。図５３に示すＤＶＢ－ＨＴＭＬデータ内には、他のＤＶＢ－ＨＴＭＬデータへのリンクが定義されている。

図５６において、ディスプレイ上の文字列“Ｊａｐａｎ”として代表されるリンク先「ｄｖｂ：／／１．２．１／Ｊａｐａｎ＿Ｗｅａｔｈｅｒ．ｈｔｍｌ」のクリックをユーザが入力すると、インタラクション部５２０４は、ＤＶＢ－ＨＴＭＬデータの情報「ｄｖｂ：／／１．２．１／Ｊａｐａｎ＿Ｗｅａｔｈｅｒ．ｈｔｍｌ」をＨＴＭＬマネージャ５２０５に引き渡す。ここで、あるリンク先のクリックを入力するとは、そのリンク先をフォーカス５６０１が選択しているときにＯＫボタン３０５を押下することをいう。ＨＴＭＬマネージャ５２０５は、ＤＶＢ－ＨＴＭＬデータの情報「ｄｖｂ：／／１．２．１／Ｊａｐａｎ＿Ｗｅａｔｈｅｒ．ｈｔｍｌ」に基づいて、ＤＶＢ－ＨＴＭＬ

データをダウンロードする。そして、ＨＴＭＬマネージャ５２０５は、パーザー５２０１、レイアウター５２０２、描画部５２０３を用いてＤＶＢ－ＨＴＭＬデータを解釈・表示する。図５７は、ユーザが文字列“Ｊａｐａｎ”と代表されるリンク先をクリックした後にディスプレイ表示されるＤＶＢ－ＨＴＭＬデータの例を示す図である。

また、例えば、ＤＶＢ－ＨＴＭＬデータ内に定義されたボタンに、ＤＯＭツリーの構成を変更するスクリプトが定義されているとする。ここで、ユーザがボタンのクリックを入力部１１１から入力すると、インタラクション部５２０４は、パーザー５２０１からＤＶＢ－ＨＴＭＬデータを受け取り、ボタンに定義されたスクリプトを実行し、ＤＯＭツリーの構成を変更する。この変更に伴い、レイアウター５２０２が表示位置を再計算し、描画部５２０３は描画をやり直す。また、ＤＯＭツリーを変更したことを、ＨＴＭＬマネージャ５２０５に通知する。

図５８に示すＤＶＢ－ＨＴＭＬデータ内には、ＤＯＭツリーの構成を変更するスクリプト５８０１として”ｃｈａｎｇｅＤＯＭＴｒｅｅ（）”（具体的には、例えば、ＤＯＭツリーにおいてテーブル要素を記述したもの）が定義され、ボタン５８０２のクリック時にそのスクリプト５８０１を実行するよう定義されている。スクリプト５８０１において、ＤＯＭツリーの構成を変更するソースコードは省略されている。スクリプトは、例えば、ＤＶＢ－ＭＨＰが規定するＤＯＭ（Ｗ３Ｃが規定するＤＯＭ２の拡張仕様）仕様に従うものであっても、ＤＯＭ１、２等の他の仕様に

従うスクリプトであっても実施可能である。

図５９は、図５８に示すＤＶＢ－ＨＴＭＬデータに対してパーザー５２０１が構築したＤＯＭツリーの一部を示す図である。また、図６０は、レイアウター５２０２および描画部５２０３によって表示されたディスプレイ表示の例を示す図である。図６０において、ラベル“詳細情報”６００１に代表されるボタン５８０２のクリックをユーザが入力すると、インタラクション部５２０４は、パーザー５２０１からＤＶＢ－ＨＴＭＬデータを受け取る。そして、インタラクション部５２０４は、ボタン５８０２に定義されたスクリプト５８０１を実行してＤＯＭツリーの構成を変更する。また、レイアウター５２０２は、表示位置を最計算する。描画部５２０３は、描画をやり直す。さらに、ＤＯＭツリーを変更したことがＨＴＭＬマネージャ５２０５に通知される。図６１は、ユーザによるボタンのクリックに伴うＤＯＭツリーの変更結果を示す図である。また、図６２は、ディスプレイ表示の変更結果の例を示す図である。

ＨＴＭＬマネージャ５２０５は、放送波中に多重化されているＤＶＢ－ＨＴＭＬデータをダウンロードする。そして、ＨＴＭＬマネージャ５２０５は、パーザー５２０１、レイアウター５２０２、および描画部５２０３を用いてＤＶＢ－ＨＴＭＬデータを解釈し、表示する。ＭＨＰ１．１規格によれば、放送波中には、ＡＩＴと呼ばれるＤＶＢ－ＨＴＭＬデータを定義する表が多重化され伝送される。ＨＴＭＬマネージャ５２０５は、このＡＩＴを最初にダウン

ロードする。ＡＩＴには、ＤＶＢ－ＨＴＭＬデータをダウンロードするための情報や、ＤＶＢ－ＨＴＭＬデータの名前、ＤＶＢ－ＨＴＭＬデータの制御情報などが含まれる。

図６３は、ＡＩＴの主要部を表した模式図である。図６３において最も左側の列には、ＤＶＢ－ＨＴＭＬデータのＩＤが記述されている。左から２番目の列には、ＤＶＢ－ＨＴＭＬデータの制御情報が記述されている。制御情報には「ａｕｔｏｓｔａｒｔ」、「ｐｒｅｓｅｎｔ」、「ｋｉｌｌ」等がある。「ａｕｔｏｓｔａｒｔ」は、ＨＴＭＬブラウザ７５０がＤＶＢ－ＨＴＭＬデータを即時に解釈・表示することを意味する。「ｐｒｅｓｅｎｔ」は、自動的に解釈・表示しないことを意味する。「ｋｉｌｌ」は、ＤＶＢ－ＨＴＭＬデータの表示を消すことを意味する。左から３番目の列には、ＤＶＢ－ＨＴＭＬデータのデータ名が記述されている。最も右側の列には、ＤＶＢ－ＨＴＭＬデータの優先度が記述されている。複数のＤＶＢ－ＨＴＭＬデータの表示時に、リソースの競合が発生した場合、優先度の高いＤＶＢ－ＨＴＭＬデータの表示にリソースが優先的に割り当てられる。ここで、優先度の値が大きいほど優先度が高いこととする。

図６３において、上から２番目および３番目の行には、ＤＶＢ－ＨＴＭＬデータの情報の組が記述される。上から２番目の行において定義されるＤＶＢ－ＨＴＭＬデータについては、ＩＤが「１」であり、制御情報が「ａｕｔｏｓｔａｒｔ」であり、データ名が「天気予報」であり、優先度が「５４」である。上から３番目の行において定義され

るＤＶＢ－ＨＴＭＬデータについては、ＩＤが「２」であり、制御情報が「ｐｒｅｓｅｎｔ」であり、データ名が「レジャー情報」であり、優先度が「２２」である。

ＨＴＭＬマネージャ５２０５は、図６３に示すＡＩＴを受け取ると、データ名が「天気予報」であるＤＶＢ－ＨＴＭＬデータをダウンロードし、解釈・表示する。ここで、ＤＶＢ－ＭＨＰ１．１規格が規定するＤＶＢ－ＨＴＭＬデータは、「Ｌｏａｄｅｄ」、「Ｐａｕｓｅｄ」、「Ａｃｔｉｖｅ」、「Ｄｅｓｔｒｏｙｅｄ」、および「Ｋｉｌｌｅｄ」という５つの状態を有している。

図６４は、ＤＶＢ－ＨＴＭＬデータの５つの状態および状態間の遷移を表す状態遷移図である。ＨＴＭＬマネージャ５２０５がＤＶＢ－ＨＴＭＬデータをダウンロードし、１次記憶部１０９に格納されると、当該ＤＶＢ－ＨＴＭＬデータの状態は「Ｌｏａｄｅｄ」になる。各状態間を結ぶ矢印は、状態間で可能な遷移を表しており、矢印のない状態間の遷移は発生しない。例えば、「Ｄｅｓｔｒｏｙｅｄ」の状態から「Ａｃｔｉｖｅ」の状態に遷移することはない。

ＨＴＭＬマネージャ５２０５は、ＤＶＢ－ＨＴＭＬデータの状態遷移を、（１）放送波中のＡＩＴの制御情報、（２）ＤＶＢ－ＨＴＭＬデータの指示、および（３）ナビゲータ７２０が結合部７３０を通して送る指示、の３通りの指示に従って行う。

ＡＩＴの制御情報が「ａｕｔｏｓｔａｒｔ」の場合、ＨＴＭＬマネージャ５２０５は、ＤＶＢ－ＨＴＭＬデータを

ダウンロードした後、パーザー３６０２、レイアウター３６０３、および描画部３６０４に指示することでＤＶＢ－ＨＴＭＬデータを解釈・表示し、「Ａｃｔｉｖｅ」状態に遷移させる。ＡＩＴは、時間と共に変化する。例えば、ＡＩＴが図６３に示す状態から図６５に示す状態へ変化したとする。図６５では、データ名が「天気予報」であるＤＶＢ－ＨＴＭＬデータの制御情報が「ａｕｔｏｓｔａｒｔ」から「ｋｉｌｌ」に変更されている。このとき、ＨＴＭＬマネージャ５２０５は、ＤＶＢ－ＨＴＭＬデータの表示状態を「Ｋｉｌｌｅｄ」という状態に遷移させる。

一方、ＤＶＢ－ＨＴＭＬデータに対するユーザの入力により、ＤＶＢ－ＨＴＭＬデータが状態遷移を発生させることができる。図６６は、ＤＶＢ－ＨＴＭＬデータ内に他のＤＶＢ－ＨＴＭＬデータへのリンクが定義されている例を示す図である。図６７は、図６６に示すＤＶＢ－ＨＴＭＬデータの表示例を示す図である。図６７において、表示６７１１がリンク６６０１に対応する。ここで、ユーザがリンク６６０１に対してクリックを入力部１１１から入力すると、インタラクション部５２０４は、リンクに定義されたＤＶＢ－ＨＴＭＬデータの情報をＨＴＭＬマネージャ５２０５に引き渡す。ここで、ＤＶＢ－ＨＴＭＬデータの情報は図６６を参照して「ｄｖｂ：／／１．２．１／Ｊａｐａｎ＿Ｗｅａｔｈｅｒ．ｈｔｍｌ」である。ＨＴＭＬマネージャ５２０５は、引き渡されたＤＶＢ－ＨＴＭＬデータの情報に基づいて、ＤＶＢ－ＨＴＭＬデータをダウンロードする。そして、ＨＴＭＬマネージャ５２０５は、パーザ

ー５２０１、レイアウター５２０２、および描画部５２０３を用いてＤＶＢ－ＨＴＭＬデータを解釈・表示する。この処理によって、ＨＴＭＬマネージャは、元のＤＶＢ－ＨＴＭＬデータの状態を「Ａｃｔｉｖｅ」から「Ｋｉｌｌｅｄ」へと遷移することになる。一方、「ｄｖｂ：／／１．２．１／Ｊａｐａｎ＿Ｗｅａｔｈｅｒ．ｈｔｍｌ」という情報に相当する新たなＤＶＢ－ＨＴＭＬデータの状態は、「Ｌｏａｄｉｎｇ」から「Ａｃｔｉｖｅ」へと遷移することになる。図６８は、情報「ｄｖｂ：／／１．２．１／Ｊａｐａｎ＿Ｗｅａｔｈｅｒ．ｈｔｍｌ」に相当する新たなＤＶＢ－ＨＴＭＬデータの表示例を示す図である。

最後に、ナビゲータ７２０からの指示によってＤＶＢ－ＨＴＭＬデータの表示の状態を遷移する方法を説明する。ナビゲータ７２０は、共通状態、すなわち、結合部７３０が規定しているフォーマットエンジンの状態およびフォーマットエンジンが表示可能なデータの状態に基づいて指示を出す。この共通状態は、個別状態、すなわち、ＨＴＭＬブラウザ７５０のＨＴＭＬマネージャ５２０５が表示するＤＶＢ－ＨＴＭＬデータの表示の状態および状態遷移とは異なっている。この共通状態と個別状態との間のギャップを第２の変換部７５１が埋めるのである。第２の変換部７５１は、ナビゲータ７２０からの指示を、ＨＴＭＬブラウザ７５０が取り扱う状態および状態遷移に変換する。その結果、ＨＴＭＬマネージャ５２０５は、ＡＩＴの制御情報やＤＶＢ－ＨＴＭＬデータからの指示と同等の指示を受け、ＤＶＢ－ＨＴＭＬデータの表示の状態を遷移させること

ができる。

第２の変換部７５１は、結合部７３０からのメッセージを変換してＨＴＭＬブラウザ７５０に伝え、また、ＨＴＭＬブラウザ７５０の情報を変換して結合部７３０に伝えることにより、ＨＴＭＬブラウザ７５０の動作を結合部７３０が規定する状態に適合させる。ここまで、様々なメッセージを、結合部７３０を通してＨＴＭＬブラウザ７５０に配送すると記述してきたが、正確には、これらのメッセージはＨＴＭＬブラウザ７５０ではなく第２の変換部７５１に送信される。第２の変換部７５１が結合部７３０から「フォーマットエンジン状態要求」メッセージを受け取った場合、第２の変換部７５１は、ＨＴＭＬブラウザ７５０が表示可能な全てのＨＴＭＬデータの状態に基づいてＨＴＭＬブラウザ７５０の状態を決定する。そして、第２の変換部７５１は、「フォーマットエンジン状態応答」メッセージを作成し、結合部７３０へ送る。

図６９は、ＨＴＭＬブラウザ７５０が表示可能な全てのＨＴＭＬデータの状態と、ＨＴＭＬブラウザ７５０の動作状態との対応を示す変換表の一例を示す図である。図６９において左側の列には、ＨＴＭＬブラウザ７５０が表示可能な全てのＨＴＭＬデータの状態が記述されている。右側の列には、対応するＨＴＭＬブラウザ７５０の動作状態が記述されている。ここで、ＨＴＭＬブラウザ７５０の状態は、結合部７３０が規定した状態、すなわち共通状態である。具体的には、図２１で示した「実行中」、「一時停止中」、および「停止中」という３つの動作状態である。上

から２番目から４番目の行には、これら３つの状態に対応する、ＨＴＭＬブラウザ７５０が表示可能な全てのＨＴＭＬデータの状態が規定されている。

まず、上から２番目の行には、全てのＨＴＭＬデータの中で１つでも「Ａｃｔｉｖｅ」状態のＨＴＭＬデータがあることに対して、ＨＴＭＬブラウザ７５０の共通状態として「実行中」が対応付けられている。一般に、「Ａｃｔｉｖｅ」状態のＨＴＭＬデータの表示およびユーザとのインタラクションには極小リソースを使用している可能性が高いため、このような対応関係を規定した。

次に、上から３番目の行には、「Ａｃｔｉｖｅ」状態のＨＴＭＬデータがなく、かつ、全ＨＴＭＬデータの中で１つでも「Ｐａｕｓｅｄ」状態のＨＴＭＬデータがあることに対して、ＨＴＭＬブラウザ７５０の共通状態として「一時停止中」が対応付けられている。ＤＶＢ－ＭＨＰ１．１規格によれば、「Ｐａｕｓｅｄ」状態のＨＴＭＬデータの表示は、一部のリソースは使用できないと記述されている。よって、このような対応関係を規定した。

次に、上から４番目の行には、上から２番目および３番目の行に示す場合以外の場合に対して、「停止中」が対応付けられている。上から２番目および３番目の行に示す場合以外の場合とは、具体的には、全てのＨＴＭＬデータが「Ｌｏａｄｅｄ」、「Ｄｅｓｔｒｏｙｅｄ」、もしくは「Ｋｉｌｌｅｄ」状態、または読み込みも完了していない状態にある場合である。この場合、ＨＴＭＬデータに対するインタラクション処理は実行されていないので、リソース

を一切使用していない「停止中」に対応付けるのが妥当である。

前述した通り、ＨＴＭＬブラウザ７５０のＨＴＭＬマネージャ５２０５は、ＡＩＴの制御情報の変化やＨＴＭＬデータの指示により、ＨＴＭＬデータの状態を遷移させる。第２の変換部７５１は、状態変化の通知をＨＴＭＬマネージャ５２０５より受ける。その結果としてＨＴＭＬブラウザ７５０の状態が変化する場合には、第２の変換部７５１は、図６９の変換表を参照して、「フォーマットエンジン状態変化」メッセージを生成して、結合部７３０に通知する。この場合、第２の変換部７５１は、１次記憶部１０９にＨＴＭＬブラウザ７５０の共通状態を保存しておく。そして、ＨＴＭＬデータの状態が変化した後の共通状態と比較する。なお、第２の変換部７５１は、ＨＴＭＬデータの状態の変化をＨＴＭＬマネージャ５２０５から通知されたとき、常に「フォーマットエンジン状態変化」メッセージを生成して、結合部７３０に通知してもよい。

第２の変換部７５１が結合部７３０から「フォーマットエンジン実行」メッセージを受け取った場合、第２の変換部７５１は、ＡＩＴに規定された所定のＨＴＭＬデータを「Ａｃｔｉｖｅ」状態に遷移させるようにＨＴＭＬブラウザ７５０に指示する。ここで所定のＨＴＭＬデータとは、ＡＩＴに規定されている全てのＨＴＭＬデータでもよいし、優先度が最も高いＨＴＭＬデータであってもよい。所定のＨＴＭＬデータを「Ａｃｔｉｖｅ」状態に遷移させるには、ＤＶＢ－ＭＨＰ１．１規格が規定しているｏｒｇ．ｄ

ｖｂ．ａｐｐｌｉｃａｉｔｏｎパッケージの機能を利用することができる。状態を変化させるべきＨＴＭＬデータのＩＤを指定した後、遷移させる状態「Ａｃｔｉｖｅ」を指定することによって、対象のＨＴＭＬデータの状態を「Ａｃｔｉｖｅ」状態に遷移させることができる。なお、すでに「Ａｃｔｉｖｅ」状態のＨＴＭＬデータが存在する場合は、第２の変換部７５１は何もしないということとしてもよい。

第２の変換部７５１が結合部７３０から「フォーマットエンジン停止」メッセージを受け取った場合、第２の変換部７５１は、ＡＩＴに規定された全てのＨＴＭＬデータを「Ｋｉｌｌｅｄ」状態に遷移させるようにＨＴＭＬブラウザ７５０に指示する。ただし、読み込んでいないＨＴＭＬデータに対しては何もしなくてもよい。全てのＨＴＭＬデータを「Ｋｉｌｌｅｄ」状態に遷移させるには、ＤＶＢ－ＭＨＰ１．１規格が規定しているｏｒｇ．ｄｖｂ．ａｐｐｌｉｃａｉｔｏｎパッケージの機能を利用することができる。ＨＴＭＬデータのＩＤを指定した後、遷移させる状態「Ｋｉｌｌｅｄ」を指定することで、対象のＨＴＭＬデータの状態を「Ｋｉｌｌｅｄ」状態に遷移させることができる。この操作をＡＩＴが定義する全ＨＴＭＬデータに対して行えばよい。

第２の変換部７５１が結合部７３０から「フォーマットエンジン一時停止」メッセージを受け取った場合、第２の変換部７５１は、ＡＩＴに規定されたＨＴＭＬデータの中で「Ａｃｔｉｖｅ」状態のＨＴＭＬデータを見つけ、その

ＨＴＭＬデータの状態を「Ｐａｕｓｅｄ」状態に遷移させるようにＨＴＭＬブラウザ７５０に指示する。この処理を実現するには、ＤＶＢ－ＭＨＰ１．１規格が規定しているｏｒｇ．ｄｖｂ．ａｐｐｌｉｃａｉｔｏｎパッケージの機能を利用することができる。ＨＴＭＬデータのＩＤを指定した後、そのＨＴＭＬデータの状態を取得する。もし状態が「Ａｃｔｉｖｅ」であれば、遷移させる状態「Ｐａｕｓｅｄ」を指定することによって、対象のＨＴＭＬデータの状態を「Ｐａｕｓｅｄ」状態に遷移させることができる。この操作をＡＩＴが定義する全てのＨＴＭＬデータに対して行えばよい。

第２の変換部７５１が結合部７３０から「アプリケーション・データ一覧要求」メッセージを受け取った場合、第２の変換部７５１は、ＡＩＴが定義している全てのＨＴＭＬデータおよびその状態に基づいて、「アプリケーション・データ一覧応答」メッセージを作成し、結合部７３０に送る。「アプリケーション・データ一覧応答」メッセージに含めるアプリケーション名は、ＡＩＴが規定するアプリケーション名がそのまま使用される。「アプリケーション・データ一覧応答」メッセージに含める状態ＩＤは、各ＨＴＭＬデータの状態を結合部７３０が規定する共通状態に変換表を用いて変換することによって作成される。

図７０は、各ＨＴＭＬデータの状態を共通状態に変換するための変換表の一例を示す図である。図７０において左側の列には、ＤＶＢ－ＭＨＰ１．１規格が規定しているＨＴＭＬデータの状態が記述されている。右側の列には、対

応する共通状態が記述されている。上から２番目の行に示されるように、ＤＶＢ－ＭＨＰ１．１規格が規定しているＨＴＭＬデータの状態「Ａｃｔｉｖｅ」は、「実行中」という共通状態に対応付けられている。一般に、「Ａｃｔｉｖｅ」状態のＨＴＭＬデータの表示およびユーザとのインタラクションは極小リソースを使用する可能性が高いので、このような対応関係を規定した。

また、図７０において上から３番目の行に示されるように、ＤＶＢ－ＭＨＰ１．１規格が規定しているＨＴＭＬデータの状態「Ｐａｕｓｅｄ」は、「一時停止中」という共通状態に対応付けられている。ＤＶＢ－ＭＨＰ１．１規格によれば、「Ｐａｕｓｅｄ」状態のＨＴＭＬデータの表示およびユーザとのインタラクション処理は使用可能なリソースが制限されると記述されている。よって、このような対応関係を規定した。

また、上から４番目の行に示されるように、上から２番目および３番目の行以外の場合のＨＴＭＬデータの状態に対しては、「停止中」という共通状態が対応付けられている。上から２番目および３番目の行以外の場合とは、具体的には、ＨＴＭＬデータが「Ｌｏａｄｅｄ」、「Ｄｅｓｔｒｏｙｅｄ」、もしくは「Ｋｉｌｌｅｄ」状態、または読み込みも完了していない状態にある場合である。この場合、ＨＴＭＬデータの解釈・表示処理およびユーザとのインタラクション処理は実行されないので、リソースを一切使用していない「停止中」に対応付けるのが妥当である。

今、データ名が「天気予報」であるＨＴＭＬデータの状

態が「Ａｃｔｉｖｅ」であり、データ名が「レジャー情報」であるＨＴＭＬデータの状態が「Ｄｅｓｔｒｏｙｅｄ」であるように、ＡＩＴにおいて定義されている場合を考える。この場合、第２の変換部７５１は、「アプリケーション・データ一覧要求」メッセージを受け取ると、図７１に示す「アプリケーション・データ一覧応答」メッセージ４７０.０を生成し、結合部７３０に送信する。

図７１は、第２の変換部７５１が送信するメッセージの具体例を示す図である。図７１に示すメッセージ７１００は、ＨＴＭＬブラウザ７５０からナビゲータ７２０への「アプリケーション・データ取得応答」である。Ｓｏｕｒｃｅ　ＩＤフィールド７１０１には、ＨＴＭＬブラウザ７５０を表すサブプログラムＩＤとして「３」が格納される（図１８参照）。Ｄｅｓｔｉｎａｔｉｏｎ　ＩＤフィールド７１０２には、ナビゲータ７２０を表すサブプログラムＩＤとして「１」が格納される（図１８参照）。Ｍｅｓｓａｇｅ　ＩＤフィールド７１０３には、「アプリケーション・データ取得応答」を表すメッセージＩＤとして「１２」が格納される（図１９参照）。Ｄａｔａ　Ｌｅｎｇｔｈフィールド７１０４には、データの長さを表す「２７」が格納される。アプリケーション数フィールド７１０５には、ＨＴＭＬブラウザ７５０が受け取ったＡＩＴに規定されているＨＴＭＬデータ数である「２」が格納される。

また、図７１において、アプリケーション数フィールド７１０５には、１つ目のＨＴＭＬデータに対するアプリケーション情報フィールド７１１１と、２つ目のＨＴＭＬデ

ータに対するアプリケーション情報フィールド７１１２とが含まれている。アプリケーションＩＤフィールド７１２１には、１つ目のＨＴＭＬデータを表すアプリケーションＩＤとして「１」が格納されている。アプリケーション状態ＩＤフィールド７１２２には、「実行中」を表すアプリケーション状態ＩＤとして「３」が格納されている。この「実行中」は、ＨＴＭＬデータの状態「Ａｃｔｉｖｅ」が変換された結果である。アプリケーション名長さフィールド７１２３には、アプリケーション名の長さを示す「８」が格納される。アプリケーション名フィールド７１２４には、ＡＩＴで定義されたデータ名「天気予報」が格納されている。ここで「天気予報」の各文字は２バイトコードで表現されており、結果、アプリケーション名長さは、４文字×２バイト＝８バイトとなっている。アプリケーションＩＤフィールド７１２５には２つ目のＨＴＭＬデータを表すアプリケーションＩＤとして「２」が格納されている。状態ＩＤフィールド７１２６には、「停止中」を表すアプリケーション状態ＩＤとして「３」が格納されている。この「停止中」は、Ｊａｖａアプリケーションの動作状態「Ｄｅｓｔｒｏｙｅｄ」が変換された結果である。アプリケーション名長さフィールド７１２７には、アプリケーション名の長さを示す「１２」が格納される。アプリケーション名フィールド７１２８には、ＡＩＴで定義されたデータ名「レジャー情報」が格納されている。ここで「レジャー情報」の各文字は２バイトコードで表現されており、結果、アプリケーション名長さは、６文字×２バイト＝１２バ

イトとなっている。

前述した通り、ＨＴＭＬブラウザ７５０のＨＴＭＬマネージャ５２０５は、ＡＩＴの制御情報の変化やＨＴＭＬデータの指示によって、ＨＴＭＬデータの状態を遷移させる。第２の変換部７５１は、状態変化の通知をＨＴＭＬマネージャ５２０５より受ける。その時、第２の変換部７５１は、「アプリケーション・データ一覧変化」メッセージを生成して、結合部７３０に通知する。第２の変換部７５１が、結合部７３０から「アプリケーション・データ実行」メッセージを受け取った場合、第２の変換部７５１は、メッセージのデータフィールド１７５で指定されるＨＴＭＬデータの状態を「Ａｃｔｉｖｅ」状態に遷移させるようにＨＴＭＬマネージャ５２０５に指示する。ＨＴＭＬデータを「Ａｃｔｉｖｅ」状態に遷移させるには、ＤＶＢ－ＭＨＰ１．１規格が規定しているｏｒｇ．ｄｖｂ．ａｐｐｌｉｃａｉｔｏｎパッケージの機能を利用することができる。状態を変化させるＨＴＭＬデータのＩＤを指定した後、遷移させる状態「Ａｃｔｉｖｅ」を指定することによって、対象のＨＴＭＬデータの状態を「Ａｃｔｉｖｅ」状態に遷移させることができる。なお、すでにＨＴＭＬデータが「Ａｃｔｉｖｅ」状態の場合、第２の変換部７５１は何もしないとようにしてもよい。

第２の変換部７５１が結合部７３０から「アプリケーション・データ停止」メッセージを受け取った場合、第２の変換部７５１は、メッセージのデータフィールド１７５で指定されるＨＴＭＬデータの状態を「Ｋｉｌｌｅｄ」に遷

移させるようにＨＴＭＬマネージャ５２０５に指示する。ただし、読み込んでいないＨＴＭＬデータに対しては何もしなくてもよい。ＨＴＭＬデータを「Ｋｉｌｌｅｄ」状態に遷移させるには、ＤＶＢ－ＭＨＰ１．１規格が規定しているｏｒｇ．ｄｖｂ．ａｐｐｌｉｃａｉｔｏｎパッケージの機能を利用することができる。ＨＴＭＬデータのＩＤを指定した後、遷移させる状態「Ｋｉｌｌｅｄ」を指定することで、対象のＨＴＭＬデータの状態を「Ｋｉｌｌｅｄ」状態に遷移させることができる。

第２の変換部７５１が結合部７３０から「アプリケーション・データー時停止」メッセージを受け取った場合、第２の変換部７５１は、メッセージのデータフィールド１７５で指定されるＨＴＭＬデータの状態を「Ｐａｕｓｅｄ」に遷移させるようにＨＴＭＬマネージャ５２０５に指示する。この処理を実現するには、ＤＶＢ－ＭＨＰ１．１規格が規定しているｏｒｇ．ｄｖｂ．ａｐｐｌｉｃａｉｔｏｎパッケージの機能を利用することができる。ＨＴＭＬデータのＩＤを指定した後、遷移させる状態「Ｐａｕｓｅｄ」を指定することで、対象のＨＴＭＬデータの状態を「Ｐａｕｓｅｄ」状態に遷移させることができる。

さらに、第２の変換部７５１は、結合部７３０のリソース管理部７３３よりリソース剥奪の通知を受けてもよい。第２の変換部７５１は、剥奪されたリソースに応じて、ＨＴＭＬマネージャ５２０５にＨＴＭＬデータの状態を遷移させることとしてもよい。ＨＴＭＬデータの表示に不可欠なリソースが剥奪された場合は、全てのＨＴＭＬデータの

状態を「Ｋｉｌｌｅｄ」に遷移させる。それ以外のリースの剥奪に対しては、第２の変換部７５１は、何も行わないとしてもよい。

また、リソース剥奪通知は、ＨＴＭＬブラウザ７５０が直接受け取り、対応する処理を行ってもよい。以上で、ＨＴＭＬブラウザ７５０の説明を終了する。

次に、図７２および図７３を用いて、メーラー７６０について説明する。メーラー７６０は、電子メールの読み書きを行うためのフォーマットエンジンである。メーラー７６０は、Ｊａｖａミドルウェア７４０やＨＴＭＬブラウザ７５０のように、取り扱うアプリケーションやデータに関する状態を規定していない。メーラー７６０自身の状態も特に規定がない。ここでは、メーラー７６０は、実行中および停止中という２つの状態を有している考える。ここで、メーラー７６０は一般的な電子メールソフトウエアの機能を含む。メーラー７６０自体の機能は既知であるので説明を省略する。

第３の変換部７６１は、結合部７３０からのメッセージを変換してメーラー７６０に伝える。また、第３の変換部７６１は、メーラー７６０の動作状態を変換して結合部７３０に伝える。これによって、第３の変換部７６１は、メーラー７６０の動作状態を結合部７３０が規定する共通状態に適合させる。なお、ここまで、様々なメッセージを、結合部７３０を通してメーラー７６０に配送すると記述してきたが、正確には、これらのメッセージはメーラー７６０ではなく第３の変換部７６１に配送される。

第３の変換部７６１が結合部７３０から「フォーマットエンジン状態要求」メッセージを受け取った場合、第３の変換部７６１は、メーラー７６０の動作状態に基づいて「フォーマットエンジン状態応答」メッセージを作成し、結合部７３０に送る。

図７２は、メーラー７６０の状態と、結合部７３０が規定している共通状態との対応を示す変換表の一例を示す図である。図７２において左側の列には、メーラー７６０の個別状態が記述されている。右側の列には、対応するメーラー７６０の共通状態が記述されている。共通状態は、具体的には、図２１で示した「実行中」および「停止中」という２つの動作状態である。上から２番目および３番目の行には、これら２つの状態に対応する、メーラー７６０の個別状態が規定されている。

まず、上から２番目の行には、メーラー７６０が起動中であることに対して、メーラー７６０の共通状態として「実行中」が対応付けられている。一般に、起動中のメーラーはネットワークインターフェース等の極小リソースを使用している可能性が高いため、このような対応関係を規定した。

次に、上から３番目の行には、メーラー７６０が停止中であることに対して、メーラー７６０の共通状態として「停止中」が対応付けられている。メーラー７６０が停止しているときは、メーラー７６０はリソースを一切使用していない。よって、「停止中」に対応付けるのが妥当である。なお、メーラー７６０には、共通状態「一時停止中」に

対応する個別状態はない。

メーラー７６０は、不測の事態などが発生すると、メーラー７６０自身を終了する可能性がある。第３の変換部７６１は、この変化をメーラー７６０より受ける。このとき、第３の変換部７６１は、「フォーマットエンジン状態変化」メッセージを生成して、結合部７３０に通知する。第３の変換部７６１が結合部７３０から「フォーマットエンジン実行」メッセージを受け取った場合、第３の変換部７６１は、メーラー７６０を起動する。第３の変換部７６１が結合部７３０から「フォーマットエンジン停止」メッセージを受け取った場合、第３の変換部７６１は、メーラー７６０を終了させる。第３の変換部７６１が結合部７３０から「フォーマットエンジン一時停止」メッセージを受け取った場合、第３の変換部７６１は、メーラー７６０を終了させる。図７２で示すように、メーラー７６０には「一時停止中」に対応する状態がない。ここでは、「一時停止中」とは、極小リソースを使用しないという意味である（図２１参照）。これを実現するには、メーラー７６０を終了させるしかない。

第３の変換部７６１が結合部７３０から「アプリケーション・データ一覧要求」メッセージを受け取った場合、「アプリケーション・データ一覧応答」メッセージを作成し、結合部７３０に送る。ここで、メーラー７６０は、取り扱うアプリケーションやデータがないため、アプリケーション数０のメッセージを生成する。

図７３は、第３の変換部７６１が生成する「アプリケー

ション・データ一覧応答」メッセージの具体例を示す図である。図７３に示すメッセージ７３００は、メーラー７６０からナビゲータ７２０への「アプリケーション・データ取得応答」である。Ｓｏｕｒｃｅ　ＩＤフィールド７３０１には、メーラー７６０を表すサブプログラムＩＤとして「４」が格納される。Ｄｅｓｔｉｎａｔｉｏｎ　ＩＤフィールド７３０２には、ナビゲータ７２０を表すサブプログラムＩＤとして「１」が格納される。Ｍｅｓｓａｇｅ　ＩＤフィールド７３０３には、「アプリケーション・データ取得応答」を表すメッセージＩＤとして「１２」が格納される。Ｄａｔａ　Ｌｅｎｇｔｈフィールド７３０４には、データの長さを示す「１」が格納される。アプリケーション数フィールド７３０５には、「０」が格納される。

第３の変換部７６１が結合部７３０から「アプリケーション・データ実行」メッセージ、「アプリケーション・データ停止」メッセージあるいは「アプリケーション・データ一時停止」メッセージを受け取った場合、第３の変換部７６１は、何もしない。この理由は、メーラー７６０は、取り扱うアプリケーション・データの状態を持たないからである。また、第３の変換部７６１は、「アプリケーション・データ一覧変化」メッセージを生成することはない。

さらに第３の変換部７６１は、結合部７３０のリソース管理部７３３よりリソース剥奪の通知を受けてもよい。第３の変換部７６１は、剥奪されたリソースに応じて、メーラー７６０を終了させることとしてもよい。メーラーの実行に不可欠なリソース、例えばネットワークインターフェ

ース等が剥奪された場合は、第３の変換部７６１は、メーラー７６０を終了させる。それ以外のリースの剥奪に対しては、第３の変換部７６１は、何も行わないとしてもよい。

また、リソース剥奪通知は、メーラー７６０が直接受け取り、対応する処理を行ってもよい。以上で、メーラー７６０の説明を終了する。また、以上で、情報処理装置の具体的な構成の説明を終了する。

（詳細な動作）

次に、図７４～図８４を用いて、上記の構成例に示したデジタルテレビにおいて前述した第１および第２の動作例の動作を行う場合について説明する。

（第１の動作例の詳細な動作）

まず、図７４～図８１を用いて、第１の動作例の詳細な動作を説明する。図７４は、第１の動作例の動作を行う場合におけるデジタルテレビにおける処理の流れを示すフローチャートである。図７４において、まず、ナビゲータ７２０は、フォーマットエンジンの起動要求を受け付ける（ステップＳ１）。ここで、起動要求は、典型的にはユーザによって指示される。例えば図８に示す状態において、ユーザが入力部１１１のＯＫボタン３０５を押下することによって、Ｊａｖａミドルウェア７４０に関する起動要求が指示される。

次に、ナビゲータ７２０は、各フォーマットエンジンの内、１つを選択する（ステップＳ２）。ステップＳ２で選択したフォーマットエンジンについて、以降のステップＳ

３～Ｓ６の処理が行われる。続いて、ナビゲータ７２０は、ステップＳ２で選択したフォーマットエンジンに対する「フォーマットエンジン（図では、“ＦＭ”と記載する。）状態要求」メッセージを結合部７３０へ送信する（ステップＳ３）。このメッセージは、結合部７３０によって当該フォーマットエンジンに対応する変換部へ送信される。

ここで、結合部７３０の通信部７３１がメッセージを送信する処理について説明する。図７５は、通信部７３１が行うメッセージ送信の処理の流れを示すフローチャートである。通信部７３１は、メッセージを受け取ると（ステップＳ７５０１）、メッセージ中のＤｅｓｔｉｎａｔｉｏｎ　ＩＤフィールドの値を参照する。Ｄｅｓｔｉｎａｔｉｏｎ　ＩＤフィールドの値が「０」のとき（ステップＳ７５０２）、通信部７３１は、ナビゲータ７２０、Ｊａｖａミドルウェア７４０、ＨＴＭＬブラウザ７７５、メーラー７６０、および結合部７３０の状態管理部７３２にメッセージを送信する（ステップＳ７５０３）。Ｄｅｓｔｉｎａｔｉｏｎ　ＩＤフィールドの値が「１」のとき（ステップＳ７５０４）、通信部７３１は、ナビゲータ７２０にメッセージを送信する（ステップＳ７５０５）。Ｄｅｓｔｉｎａｔｉｏｎ　ＩＤフィールドの値が「２」のとき（ステップＳ７５０６）、通信部７３１は、Ｊａｖａミドルウェア７４０にメッセージを送信する（ステップＳ７５０７）。Ｄｅｓｔｉｎａｔｉｏｎ　ＩＤフィールドの値が「３」のとき（ステップＳ７５０８）、通信部７３１は、ＨＴＭＬブラウザ７７５にメッセージを送信する（ステップＳ７５０

９）。Ｄｅｓｔｉｎａｔｉｏｎ　ＩＤフィールドの値が「４」のとき（ステップＳ７５１０）、通信部７３１は、メーラー７６０にメッセージを送信する（ステップＳ７５１１）。Ｄｅｓｔｉｎａｔｉｏｎ　ＩＤフィールドの値が「５」のとき（ステップＳ７５１２）、通信部７３１は、結合部７３０の状態管理部７３２にメッセージを送信する（ステップＳ７５１３）。以上の処理によって、各サブプログラムからのメッセージは、適切なサブプログラムへ送信されることとなる。通信部７３１は、メッセージを受け取る度に、上記処理を行う。

次に、ステップＳ３の「フォーマットエンジン状態要求」メッセージを受信した際の各変換部の処理の流れについて説明する。

図７６は、第１の変換部７４１が「フォーマットエンジン状態要求」メッセージを受け取ったときの処理の流れを示すフローチャートである。第１の変換部７４１が「フォーマットエンジン状態要求」メッセージを受け取ると（ステップＳ７６０１）、第１の変換部７４１は、「Ａｃｔｉｖｅ」状態のＪａｖａアプリケーションがあるかどうかを調べる（ステップＳ７６０２）。「Ａｃｔｉｖｅ」状態のＪａｖａアプリケーションがある場合、第１の変換部７４１は、Ｊａｖａミドルウェア７４０が「実行中」であることを示す「フォーマットエンジン状態応答」メッセージを作成する（ステップＳ７６０３）。一方、「Ａｃｔｉｖｅ」状態のＪａｖａアプリケーションがない場合、第１の変換部７４１は、「Ｐａｕｓｅｄ」状態のＪａｖａアプリケ

ーションがあるかどうかを調べる（ステップＳ７６０４）。「Ｐａｕｓｅｄ」状態のＪａｖａアプリケーションがある場合、第１の変換部７４１は、Ｊａｖａミドルウェア７４０が「一時停止中」であることを示す「フォーマットエンジン状態応答」メッセージを作成する（ステップＳ７６０５）。一方、「Ｐａｕｓｅｄ」状態のＪａｖａアプリケーションがない場合、第１の変換部７４１は、Ｊａｖａミドルウェア７４０が「停止中」である「フォーマットエンジン状態応答」メッセージを作成する（ステップＳ７６０６）。ステップＳ７６０３、Ｓ７６０５、またはＳ７６０６の後、第１の変換部７４１は、作成した「フォーマットエンジン状態応答」メッセージを結合部７３０に送信する（ステップＳ７６０７）。

なお、「フォーマットエンジン状態要求」メッセージを受け取った場合の第２の変換部７５１における処理の流れは、図７６のフローチャートに準ずる。

図７７は、第３の変換部７７１が「フォーマットエンジン状態要求」メッセージを受け取ったときの処理の流れを示すフローチャートである。第３の変換部７７１が「フォーマットエンジン状態要求」メッセージを受け取ると（ステップＳ７７０１）、第３の変換部７７１は、メーラー７７０が起動しているかどうかを調べる（ステップＳ７７０２）。メーラーが起動している場合、第３の変換部７７１は、メーラー７７０が「実行中」であることを示す「フォーマットエンジン状態応答」メッセージを作成する（ステップＳ７７０３）。一方、メーラーが起動していない場合

、第３の変換部７７１は、メーラー７７０が「停止中」であることを示す「フォーマットエンジン状態応答」メッセージを作成する（ステップＳ７７０４）。ステップＳ７７０３またはＳ７７０４の後、第３の変換部７７１は、作成した「フォーマットエンジン状態応答」メッセージを結合部７３０に配送する（ステップＳ７７０５）。

以上、図７６および図７７に示した処理によって、ナビゲータ７２０に「フォーマットエンジン状態応答」メッセージが返ってくる。図７４の説明に戻り、ステップＳ３の次に、ナビゲータ７２０は、「フォーマットエンジン状態応答」メッセージを受信する（ステップＳ４）。続いて、ナビゲータ7-2０は、受信したメッセージに基づいて、ステップＳ２で選択したフォーマットエンジンが実行中であるか否かを判定する（ステップＳ５）。

ステップＳ５において、フォーマットエンジンが実行中でないと判定された場合、ナビゲータ７２０は、ステップＳ６の処理をスキップし、ステップＳ７の処理を行う。一方、フォーマットエンジンが実行中であると判定された場合、ナビゲータ７２０は、当該フォーマットエンジンに対する「フォーマットエンジン停止」メッセージを結合部７３０へ送信する（ステップＳ６）。このメッセージは、当該フォーマットエンジンに対応する変換部へ結合部４０３によって送信される。

次に、ステップＳ６の「フォーマットエンジン停止」メッセージを受信した際の各変換部の処理の流れについて説明する。

図７８は、第１の変換部７４１が、「フォーマットエンジン停止」メッセージを受け取った際の処理の流れを表すフローチャートである。第１の変換部７４１が「フォーマットエンジン停止」メッセージを受け取ると（ステップＳ７８０１）、第１の変換部７４１は、ＡＩＴからＪａｖａアプリケーションを１つ選択する（ステップＳ７８０２）。続いて、第１の変換部７４１は、選択したＪａｖａアプリケーションの状態が「Ｄｅｓｔｒｏｙｅｄ」であるか、あるいは読み込む前であるかを調べる（ステップＳ７８０３）。選択したＪａｖａアプリケーションの状態が「Ｄｅｓｔｒｏｙｅｄ」であるか、あるいは読み込む前のいずれかでもない場合、第１の変換部７４１は、選んだＪａｖａアプリケーションの状態を「Ｄｅｓｔｒｏｙｅｄ」に遷移させる（ステップＳ７８０４）。選択したＪａｖａアプリケーションの状態が「Ｄｅｓｔｒｏｙｅｄ」であるか、あるいは読み込む前のいずれかである場合、第１の変換部７４１は、ステップＳ７８０４の処理をスキップし、ステップＳ７８０５の処理を行う。すなわち、第１の変換部７４１は、ステップＳ７８０２において全てのＪａｖａアプリケーションを選択したか否かを判定する。１つでも選択していないＪａｖａアプリケーションがある場合、第１の変換部７４１は、ステップＳ７８０２の処理を行う。そして、第１の変換部７４１は、ステップＳ７８０２～Ｓ７８０５の処理を繰り返す。一方、全てのＪａｖａアプリケーションを選択した場合、第１の変換部７４１は、図７８の処理を終了する。以上のように、第１の変換部７４１は、ス

テップＳ７８０２～ステップＳ７８０４までの処理を全てのＪａｖａアプリケーションに対して実施する。

図７９は、第２の変換部７５１が、「フォーマットエンジン停止」メッセージを受け取った際の処理の流れを示すフローチャートである。第２の変換部７５１が「フォーマットエンジン停止」メッセージを受け取ると（ステップＳ７９０１）、第２の変換部７５１は、ＡＩＴからＨＴＭＬデータを１つ選択する（ステップＳ７９０２）。続いて、第２の変換部７５１は、選択したＨＴＭＬデータの状態が「Ｋｉｌｌｅｄ」であるか、あるいは読み込む前であるかを調べる（ステップＳ７９０３）。選択したＨＴＭＬデータの状態が「Ｋｉｌｌｅｄ」であるか、あるいは読み込む前のいずれかでもない場合、第２の変換部７５１は、選んだＨＴＭＬデータの状態を「Ｄｅｓｔｒｏｙｅｄ」に遷移させる（ステップＳ７９０４）。選択したＨＴＭＬデータの状態が「Ｋｉｌｌｅｄ」であるか、あるいは読み込む前のいずれかである場合、第２の変換部７５１は、ステップＳ７９０４の処理をスキップし、ステップＳ７９０５の処理を行う。すなわち、第２の変換部７５１は、ステップＳ７９０２～ステップＳ７９０４までの処理を全てのＨＴＭＬデータに対して実施する（ステップＳ７９０５）。

図８０は、第３の変換部７６１が、「フォーマットエンジン停止」メッセージを受け取った際の処理の流れを示すフローチャートである。第３の変換部７６１が「フォーマットエンジン停止」メッセージを受け取ると（ステップＳ８００１）、第３の変換部７６１は、メーラー７６０が起

動しているかどうかを調べる（ステップＳ８００２）。メーラー７６０が起動していれば、第３の変換部７６１は、メーラー７６０を終了して（ステップＳ８００３）、図８０に示す処理を終了する。一方、メーラー７６０が起動していなければ、第３の変換部７６１は、ステップＳ８００３の処理をスキップして図８０に示す処理を終了する。

以上、図７８～図８０に示した処理によって、「実行中」の動作状態であったフォーマットエンジンが終了される。

図７４の説明に戻り、ステップＳ５またはステップＳ６の次に、ナビゲータ７２０は、ステップＳ２において、起動要求の対象となっているフォーマットエンジンを除く全てのフォーマットエンジンを選択したか否かを判定する（ステップＳ７）。１つでも選択していないフォーマットエンジンがある場合、ナビゲータ７２０は、ステップＳ２の処理を行う。そして、ナビゲータ７２０は、ステップＳ２～Ｓ７の処理を繰り返す。一方、全てのフォーマットエンジンを選択した場合、ナビゲータ７２０は、起動要求の対象となっているフォーマットエンジンに対する「フォーマットエンジン実行」メッセージを結合部７３０へ送信する（ステップＳ８）。その後、ナビゲータ７２０は、図７４の処理を終了する。

次に、ステップＳ８の「フォーマットエンジン実行」メッセージを受信した際の各変換部の処理の流れについて説明する。

図８１は、第１の変換部７４１が、「フォーマットエンジン実行」メッセージを受け取った際の処理の流れを示す

フローチャートである。第１の変換部７４１が「フォーマットエンジン実行」メッセージを受け取ると（ステップＳ８１０１）、第１の変換部７４１は、実行すべきＪａｖａアプリケーションを決定する（ステップＳ８１０２）。この決定は、例えば、ＡＩＴから最も優先度の高いＪａｖａアプリケーションを選ぶことによって決定される。続いて、第１の変換部７４１は、ステップＳ８１０３において決定したＪａｖａアプリケーションの動作状態が「Ａｃｔｉｖｅ」であるか否かを判定する（ステップＳ８１０３）。当該Ｊａｖａアプリケーションの状態が「Ａｃｔｉｖｅ」でなければ、第１の変換部７４１は、当該Ｊａｖａアプリケーションの状態を「Ａｃｔｉｖｅ」に遷移させ（ステップＳ８１０４）、図８１に示す処理を終了する。一方、当該Ｊａｖａアプリケーションの状態が「Ａｃｔｉｖｅ」であれば、第１の変換部７４１は、ステップＳ８１０４の処理をスキップし、図８１に示す処理を終了する。

なお、「フォーマットエンジン実行」メッセージを受け取った場合の第２の変換部７５１における処理の流れは、図８１のフローチャートに準ずる。

また、「フォーマットエンジン実行」メッセージを受け取った場合の第３の変換部７６１における処理の流れは、図８０のフローチャートに準ずる。

以上のように、図７４に示すステップＳ８において送信される「フォーマットエンジン実行」メッセージによって、起動要求の対象であるフォーマットエンジンが起動される。また、その他のフォーマットエンジン（起動要求の対

象でないフォーマットエンジン）は「実行中」の状態でなくなるので、極小リソースが競合することを回避することができる。

なお、図７４のステップＳ６では、ナビゲータ７２０は「フォーマットエンジン停止」メッセージを送信するものとしたが、ナビゲータ７２０は、「フォーマットエンジン一時停止」メッセージを送信してもよい。この場合も、上記その他のフォーマットエンジン（起動要求の対象でないフォーマットエンジン）は「実行中」の状態でなくなるので、極小リソースが競合することを回避することができる。

次に、「フォーマットエンジン一時停止」メッセージを受信した際の各変換部の処理の流れについて説明する。

図８２は、第１の変換部７４１が、「フォーマットエンジン一時停止」メッセージを受け取った際の処理の流れを表すフローチャートである。第１の変換部７４１が「フォーマットエンジン一時停止」メッセージを受け取ると（ステップＳ８２０１）、第１の変換部７４１は、ＡＩＴからＪａｖａアプリケーションを１つ選択する（ステップＳ８２０２）。続いて、第１の変換部７４１は、選択したＪａｖａアプリケーションの動作状態が「Ａｃｔｉｖｅ」であるか否かを判定する（ステップＳ８２０３）。選択したＪａｖａアプリケーションの状態が「Ａｃｔｉｖｅ」である場合、第１の変換部７４１は、選んだＪａｖａアプリケーションの状態を「Ｐａｕｓｅｄ」に遷移させる（ステップＳ８２０４）。選択したＪａｖａアプリケーションの状態

が「Ａｃｔｉｖｅ」でない場合、第１の変換部７４１は、ステップＳ８２０４の処理をスキップし、ステップＳ８２０５の処理を行う。すなわち、第１の変換部７４１は、ステップＳ８２０２～ステップＳ８２０４までの処理を全てのＪａｖａアプリケーションに対して実施する（ステップＳ８２０５）。

なお、「フォーマットエンジン一時停止」メッセージを受け取った場合の第２の変換部７５１における処理の流れは、図８２のフローチャートに準ずる。

また、「フォーマットエンジン一時停止」メッセージを受け取った場合の第３の変換部７６１における処理の流れは、「フォーマットエンジン停止」メッセージを受け取った場合と同様である。

（第２の動作例の詳細な動作）

次に、第２の動作例の詳細な動作を説明する。図８３は、第２の動作例の動作を行う場合におけるデジタルテレビにおける処理の流れを示すフローチャートである。図８３において、まず、ナビゲータ７２０は、フォーマットエンジンの起動要求を受け付ける（ステップＳ１１）。ステップＳ１１の処理は、第１の動作例におけるステップＳ１と同様である。

次に、ナビゲータ７２０は、起動要求の対象であるフォーマットエンジンに対する「フォーマットエンジン実行」メッセージを結合部７３３へ送信する（ステップＳ１２）。結合部７３３を介して当該メッセージを受け取った変換部における処理は、上述した第１の動作例と同様である。

ここで、各フォーマットエンジンは、「フォーマットエンジン実行」メッセージに応じて起動した後、必要に応じてリソースの利用をライブラリ７１１に対して要求する。リソースの利用要求を行うか否かは、各フォーマットエンジン毎に予め定められている。以下では、フォーマットエンジンが極小リソースの利用要求を行ったものとして説明する。なお、フォーマットエンジンが極小リソースの利用要求を行わない場合は、フォーマットエンジンはそのまま動作を続ける。

ステップＳ１２の次に、ライブラリ７１１は、起動したフォーマットエンジンから極小リソースの利用要求を受け付ける（ステップＳ１３）。なお、この段階では、当該フォーマットエンジンに対して当該極小リソースの利用が許可されたわけではない。続いて、ライブラリ７１１は、極小リソースの利用要求が重複しているか否かを判定する（ステップＳ１４）。ここで、「極小リソースの利用要求が重複する」とは、単一の極小リソースに対して複数の利用要求があったことを意味する。従って、異なる２つの極小リソースの各々に対して利用要求があっても、極小リソースの利用要求が重複したことにはならない。以下、ステップＳ１４における判定処理を詳細に説明する。

ステップＳ１４において、ライブラリ７１１は、起動要求を行っているフォーマットエンジンをフォーマットエンジン特定部３９０５に問い合わせる。この問い合わせに応じて、フォーマットエンジン特定部３９０５は、プロセス記憶部３９０１を参照することによって、起動要求を行っ

ているフォーマットエンジンを特定する。そして、フォーマットエンジン特定部３９０５は、特定したフォーマットエンジンをライブラリ７１１に通知する。なお、極小リソースをすでに利用しているフォーマットエンジンはすでに特定されている。従って、ライブラリ７１１は、極小リソースをすでに利用しているフォーマットエンジン（これがない場合は重複しないと判定する。）と、フォーマットエンジン特定部３９０５から通知されたフォーマットエンジンとによって、極小リソースの利用要求が重複しているか否かを判定することができる。

ステップＳ１４において、利用要求が重複していない場合、ライブラリ７１１は、利用要求を行ったフォーマットエンジンに対して極小リソースの利用を許可し（ステップＳ１９）、図８３に示す処理を終了する。

一方、ステップＳ１４において、利用要求が重複している場合、ライブラリ７１１は、優先度情報を取得する（ステップＳ１５）。具体的には、リソース管理部７３３の優先度情報提供部３９０６に対して、優先度情報の取得を要求する。ここで、優先度情報とは、優先度格納部３９０２に格納されている優先度と、最新起動記憶部３９０３に記憶されている最新起動情報とを含む概念である。つまり、優先度情報提供部３９０６は、ライブラリ７１１からの要求に対して、優先度格納部３９０２に格納されている優先度と、最新起動記憶部３９０３に記憶されている最新起動情報とを送信する。

ライブラリ７１１は、優先度情報提供部３９０６から送

信されてきた情報を用いて、利用を許可するフォーマットエンジンを決定する（ステップＳ１６）。このステップＳ１６によって、図６に示す許可決定部６１の機能が実現されることになる。なお、ここでは、ステップＳ１６の処理はライブラリが行うこととしたが、利用を許可するフォーマットエンジンを優先度情報提供部３９０６が決定し、決定したフォーマットエンジンをライブラリ７１１へ通知するようにしてもよい。

ステップＳ１６の処理は、例えば以下のように行われる。なお、ここでは、ステップＳ１３において起動要求を行ったフォーマットエンジンをフォーマットエンジンＡとし、極小リソースをすでに利用していたフォーマットエンジンをフォーマットエンジンＢと呼ぶ。

ライブラリ７１１は、まず、最新起動情報に基づいて、ユーザが最近起動したフォーマットエンジンがフォーマットエンジンＡであるか否かを判定する。そして、ユーザが最近起動したフォーマットエンジンがフォーマットエンジンＡであれば、ライブラリ７１１は、極小リソースの利用を許可するフォーマットエンジンをフォーマットエンジンＡに決定する。一方、ユーザが最近起動したフォーマットエンジンがフォーマットエンジンＡでなければ、ライブラリ７１１は、ユーザが最近起動したフォーマットエンジンがフォーマットエンジンＢであるか否かを判定する。そして、ユーザが最近起動したフォーマットエンジンがフォーマットエンジンＢであれば、ライブラリ７１１は、極小リソースの利用を許可するフォーマットエンジンをフォーマ

ットエンジンＢに決定する。すなわち、ライブラリ７１１は、極小リソースの利用を許可するフォーマットエンジンを、ユーザが最近起動したフォーマットエンジンに決定する。

さらに、ユーザが最近起動したフォーマットエンジンがフォーマットエンジンＡでもフォーマットエンジンＢでもない場合、ライブラリ７１１は、優先度情報記憶部３６０２に記憶されている情報を用いて極小リソースの利用を許可するフォーマットエンジンを決定する。すなわち、ライブラリ７１１は、極小リソースの利用を許可するフォーマットエンジンを、優先度情報記憶部３６０２に記憶されている優先度が高い方のフォーマットエンジンに決定する。

ステップＳ１６の次に、ライブラリ７１１は、ステップＳ１６において決定したフォーマットエンジンに対して極小リソースの利用を許可する（ステップＳ１７）。続いて、ライブラリ７１１は、ステップＳ１６において決定したフォーマットエンジンでない方のフォーマットエンジンに対してリソースが剥奪された旨の通知を行う（ステップＳ１８）。具体的には、ライブラリ７１１は、リソース剥奪通知を当該フォーマットエンジンに送信するように、リソース管理部７３３のリソース剥奪通知部３９０７に要求する。これに応じてリソース剥奪通知部３９０７がリソース剥奪通知を当該フォーマットエンジンに送信する。ステップＳ１８の後、ライブラリ７１１は、図８３に示す処理を終了する。

以上の処理によって、極小リソースの利用が許可されな

かったフォーマットエンジンに対応する変換部にリソース剥奪通知が送られる。変換部は、このリソース剥奪通知に応じて、以下の処理を行う。

図８４は、リソース剥奪通知を受け取った変換部における処理の流れを示すフローチャートである。なお、図８４に示す処理は、変換部が行うこととするが、フォーマットエンジンが同じ動作を行ってもよい。まず、変換部は、剥奪されたリソースのリソースＩＤをリソース剥奪通知部３９０７から受け取る（ステップＳ８４０１）。次に、変換部は、フォーマットエンジンが継続して実行可能か否かを判定する（ステップＳ８４０２）。ここで、「フォーマットエンジンが継続して実行可能」とは、「リソース（剥奪されたリソース）がなくても実行可能」という意味である。ステップＳ８４０２において、フォーマットエンジンが継続実行可能な場合、変換部は、そのまま図８４に示す処理を終了する。このとき、リソースが剥奪されたフォーマットエンジンは実行されたままである。

一方、ステップＳ８４０２において、フォーマットエンジンが継続実行不可能な場合、変換部は、一時停止状態にすれば実行可能か否かを判定する（ステップＳ８４０３）。ステップＳ８４０３において、フォーマットエンジンが継続実行不可能であれば、変換部は、フォーマットエンジンを停止する（ステップＳ８４０４）。一方、フォーマットエンジンが継続実行可能であれば、変換部は、フォーマットエンジンを一時停止する（ステップＳ８４０５）。

なお、図８４では、停止したり一時停止したりする対象

をフォーマットエンジンとしている。ここで、フォーマットエンジンが実行するアプリケーションや表示するデータ（上記のＪａｖａアプリケーションやＨＴＭＬデータ）に状態が規定されている場合、フォーマットエンジンの動作状態を変更するだけでなく、これらのアプリケーションやデータの状態を変更するようにしてもよい。なお、図８４で記載した状態は、共通状態である。変換部は、共通状態の変更をフォーマットの動作状態に応じて適切に変換し、フォーマットエンジンの状態を遷移させる。

なお、図８３では、ライブラリ７１１は、優先度情報として、優先度格納部３９０２に格納されている優先度と、最新起動記憶部３９０３に記憶されている最新起動情報とを用いてステップＳ１６における決定を行った。ここで、ライブラリ７１１は、まず最新起動情報のみを取得し、ユーザが最近起動したフォーマットエンジンが上記フォーマットエンジンＡでもフォーマットエンジンＢでもないとわかった後で、優先度格納部３９０２に格納されている優先度を取得してもよい。また、ライブラリ７１１は、優先度格納部３９０２に格納されている優先度のみを用いてステップＳ１６における決定を行ってもよい。以上で、第２の動作例の詳細な動作の説明を終了する。

（その他の動作例）

なお、上記の構成例では、図７４～図８４に示したメッセージの他にも、種々のメッセージのやり取りが行われる。以下、図７４～図８４に示したメッセージ以外のメッセージをやり取りする場合の変換部の処理を説明する。

図８５は、第１の変換部７４１がアプリケーションマネージャ４５０３からＪａｖａアプリケーションの状態変化通知を受けた際の処理の流れを示すフローチャートである。第１の変換部７４１がアプリケーションマネージャ４５０３からＪａｖａアプリケーションの状態変化の通知を受けると（ステップＳ８５０１）、第１の変換部７４１は、Ｊａ-ｖａアプリケーションの動作状態に基づいて、Ｊａｖａミドルウェア７４０の共通状態を決定する（ステップＳ８５０２）。さらに、第１の変換部７４１は、１次記憶部１０９に保存している変化前のＪａｖａミドルウェア７４０の共通状態と、ステップＳ８５０２で決定したＪａｖａミドルウェア７４０の共通状態とを比較し、変化前後で共通状態が変化したか否かを判定する（ステップＳ８５０３）。判定の結果、変化した場合、第１の変換部７４１は、「フォーマットエンジン状態変化」メッセージを作成し、結合部７３０に送信する（ステップＳ８５０４）。最後に、第１の変換部７４１は、状態が変化したＪａｖａアプリケーションに関して、「アプリケーション・データ一覧変化」メッセージを作成し、結合部７３０に送信する（ステップＳ８５０５）。

なお、第２の変換部７５１がＨＴＭＬマネージャ５２０５からＨＴＭＬデータの状態変化通知を受けた際の処理の流れは、図８５に示す処理に準ずる。

図８６は、第１の変換部７４１が「アプリケーション・データ一覧要求」メッセージを受け取った際の処理の流れを示すフローチャートである。第１の変換部７４１が「ア

プリケーション・データ一覧要求」メッセージを受け取ると（ステップＳ８６０１）、第１の変換部７４１は、ＡＩＴからＪａｖａアプリケーションの一覧を取得する（ステップＳ８６０２）。次に、第１の変換部７４１は、各Ｊａｖａアプリケーションの状態をＪａｖａミドルウェア７４０から取得する（ステップＳ８６０３）。第１の変換部７４１は、取得した各Ｊａｖａアプリケーションの状態を、アプリケーション・データの共通状態に変換する（ステップＳ８６０４）。最後に、第１の変換部７４１は、「アプリケーション・データ一覧応答」メッセージを作成し、結合部７３０に送信する（ステップＳ８６０５）。

なお、第２の変換部７５１が「アプリケーション・データ実行」メッセージを受け取った際の処理の流れは、図８６に示す処理に準ずる。

図８７は、第１の変換部７４１が、「アプリケーション・データ実行」メッセージを受け取った際の処理の流れを示すフローチャートである。第１の変換部７４１が「アプリケーション・データ実行」メッセージを受け取ると（ステップＳ８７０１）、第１の変換部７４１は、「アプリケーション・データ実行」メッセージで指定されたＪａｖａアプリケーションの動作状態が「Ａｃｔｉｖｅ」か否かを判定する（ステップＳ８７０２）。判定の結果、当該Ｊａｖａアプリケーションの状態が「Ａｃｔｉｖｅ」でない場合、第１の変換部７４１は、指定されたＪａｖａアプリケーションの動作状態を「Ａｃｔｉｖｅ」に遷移させる（ステップＳ８７０３）。一方、当該Ｊａｖａアプリケーショ

ンの動作状態が「Ａｃｔｉｖｅ」である場合、第１の変換部７４１は、図８７に示す処理を終了する。

なお、第２の変換部７５１が「アプリケーション・データ実行」メッセージを受け取った際の処理の流れは、図８７に示す処理に準ずる。

図８８は、第１の変換部７４１が、「アプリケーション・データ停止」メッセージを受け取った際の処理の流れを示すフローチャートである。第１の変換部７４１が「アプリケーション・データ停止」メッセージを受け取ると（ステップＳ８８０１）、第１の変換部７４１は、「アプリケーション・データ停止」メッセージで指定されたＪａｖａアプリケーションの動作状態が「Ｄｅｓｔｒｏｙｅｄ」であるか否かを判定する（ステップＳ８８０２）。判定の結果、当該Ｊａｖａアプリケーションの状態が「Ｄｅｓｔｒｏｙｅｄ」でない場合、第１の変換部７４１は、指定されたＪａｖａアプリケーションの動作状態を「Ｄｅｓｔｒｏｙｅｄ」に遷移させる（ステップＳ８８０３）。一方、当該Ｊａｖａアプリケーションの動作状態が「Ｄｅｓｔｒｏｙｅｄ」である場合、第１の変換部７４１は、図８８に示す処理を終了する。

なお、第２の変換部７５１が「アプリケーション・データ停止」メッセージを受け取った際の処理の流れは、図８８に示す処理に準ずる。

図８９は、第１の変換部７４１が、「アプリケーション・データー時停止」メッセージを受け取った際の処理の流れを示すフローチャートである。第１の変換部７４１が「

アプリケーション・データー時停止」メッセージを受け取ると（ステップＳ８９０１）、第１の変換部７４１は、「アプリケーション・データー時停止」メッセージで指定されたＪａｖａアプリケーションの状態が「Ａｃｔｉｖｅ」であるか否かを判定する（ステップＳ８９０２）。判定の結果、Ｊａｖａアプリケーションの状態が「Ａｃｔｉｖｅ」である場合、第１の変換部７４１は、指定されたＪａｖａアプリケーションの状態を「Ｐａｕｓｅｄ」に遷移させる（ステップＳ８９０３）。一方、当該Ｊａｖａアプリケーションの動作状態が「Ａｃｔｉｖｅ」でない場合、第１の変換部７４１は、図８９に示す処理を終了する。

なお、第２の変換部７５１が「アプリケーション・データー時停止」メッセージを受け取った際の処理の流れは、図８９に示す処理に準ずる。

図９０は、第３の変換部７６１がメーラー７６０が終了したことの通知をメーラー７６０から受けた際の処理の流れを示すフローチャートである。第３の変換部７６１がメーラー７６０が終了したことの通知をメーラー７６０から受けると（ステップＳ９００１）、第３の変換部７６１は、「フォーマットエンジン状態変化」メッセージを作成し、結合部７３０に送信する（ステップＳ９００２）。

図９１は、結合部７３０が「フォーマットエンジン状態変化」メッセージを受け取った場合の結合部７３０の状態管理部７３２の処理の流れを示すフローチャートである。まず、結合部７３０は、「フォーマットエンジン状態変化」メッセージを受け取る（ステップＳ９１０１）。次に、

結合部７３０は、受け取ったメッセージ内のＳｏｕｒｃｅＩＤフィールドを参照することによって、状態が変化したフォーマットエンジンを特定し、特定したフォーマットエンジンに対する「フォーマットエンジン状態要求」メッセージを送信する（ステップＳ９１０２）。その結果、結合部７３０の状態管理部７３２は、「フォーマットエンジン状態応答」メッセージを受け取り（ステップＳ９１０３）、フォーマットエンジンの状態を１次記憶部１０９に記憶する（ステップＳ９１０４）。

図９２は、結合部７３０が「アプリケーション・データ一覧変化」メッセージを受け取った場合の結合部７３０の状態管理部７３２の処理の流れを示すフローチャートである。まず、結合部７３０は、「アプリケーション・データ一覧変化」メッセージを受け取る（ステップＳ９２０１）。次に、結合部７３０は、受け取ったメッセージ内のＳｏｕｒｃｅ　ＩＤフィールドを参照することによって、状態が変化したアプリケーションを実行またはデータを表示しているフォーマットエンジンを特定し、特定したフォーマットエンジンに対する「アプリケーション・データ一覧要求」メッセージを送る（ステップＳ９２０２）。その結果、結合部７３０の状態管理部７３２は、「アプリケーション・データ一覧応答」メッセージを受け取り（ステップＳ９２０３）、アプリケーション・データの状態を１次記憶部１０９に記憶する。

なお、以上において、ナビゲータ７２０が状態管理部７３２の機能を有することによって、状態管理部７３２を結

合部７３０から削除してもよい。このような構成にすると、ナビゲータ７２０が全てのフォーマットエンジンの状態を管理することになる。

また、本構成例にいて、各フォーマットエンジンは、送られてきたメッセージの発信元に対して、何ら制限をかけない。従って、フォーマットエンジンは、他のフォーマットエンジンの動作状態や、フォーマットエンジンが実行しているアプリケーションや、表示しているデータの状態を変化させることが可能である。例えば、Ｊａｖａミドルウェア７４０が、ＨＴＭＬブラウザ７５０に対して、「フォーマットエンジン実行」メッセージを送信することによって、ＨＴＭＬブラウザ７５０は停止させることができることは明らかである。

また、本構成例では、ＲＯＭ１１０が保存する内容を２次記憶部１０８が保存することで、ＲＯＭ１１０を用いない構成とすることも可能である。また、２次記憶部１０８は、複数のサブ２次記憶部で構成し、個々のサブ２次記憶部が異なる情報を保存する構成としてもよい。例えば、あるサブ２次記憶部は状態管理部７３２が記憶するフォーマットエンジンの状態を記憶し、別のサブ２次記憶部は第１の変換部７４１が記憶するＪａｖａミドルウェア７４０の状態を記憶し、さらに別のサブ２次記憶部は、第２の変換部７５１が記憶するＨＴＭＬブラウザ７５０の状態を記憶するようにしてもよい。このように、各記憶部を詳細に分割することが可能である。

以上のように、本実施形態によれば、複数のフォーマッ

トエンジンを結合する際、結合部７３０とフォーマットエンジンとを繋ぐ変換部を用いることによって、個々のフォーマットエンジンの実装を大きく改造する必要がない。そのため、少ない工数で複数のフォーマットエンジンを結合できる。

また、結合部７３０が規定する共通状態を用いることによって、結合するフォーマットエンジンが増えた場合であっても、ナビゲータ７２０は、メッセージの送り先を増やすだけよく、追加されたフォーマットエンジンに容易に対応することができる。

また、共通状態を用いることによって、ナビゲータ７２０および結合部７３０は、複数のフォーマットエンジンの状態を容易に把握できる。従って、複数のフォーマットエンジンの管理を容易に行うことができる。具体的には、複数のフォーマットエンジンを同時に制御することが容易になる。

図９３は、フォーマットエンジンを実行しても、他のフォーマットエンジンに大きな影響を与えない状況の組み合わせを示す図である。図９３に示すように、全てのフォーマットエンジンが停止中あるいは一時停止中ということは、極小リソースを使用しているフォーマットエンジンがないことを意味する。従って、図９３に示す状況にある場合、任意のフォーマットエンジンを起動することができる。例えば図９３に示す状況においては、Ｊａｖａミドルウェア７４０およびＨＴＭＬブラウザ７５０を停止することなく、メーラー７６０を実行することができる。

なお、結合部７３０において共通状態が規定されていない場合、ナビゲータ７２０は、個々のフォーマットエンジンが規定する独自の状態の組み合わせを把握する必要があることになる。従って、この場合、ナビゲータ７２０は、同時実行が可能な状況を判断することが非常に困難となる。特に、本構成例ではＪａｖａミドルウェア７４０およびＨＴ-ＭＬブラウザ４５０については、実行するアプリケーションおよび表示するデータの状態は規定されているが、フォーマットエンジンとしての状態は規定されていない。従って、上記の場合には、ナビゲータ７２０は、Ｊａｖａミドルウェア７４０が実行するＪａｖａアプリケーションおよびＨＴＭＬブラウザ４５０が表示するデータが取り得る全ての状態の組み合わせを把握する必要がある。また、本構成例で取り上げたＪａｖａミドルウェア７４０、ＨＴＭＬブラウザ７５０、およびメーラー７６０以外のフォーマットエンジンがさらに追加される場合には、追加するフォーマットエンジンが規定する状態の組み合わせをさらに考慮する必要があるので、ますます、複数のフォーマットエンジンの管理が困難となる。

産業上の利用可能性

以上においてはデジタルテレビを実施の形態の一例として記載したが、本発明は、デジタルテレビに限らずパーソナルコンピュータや携帯電話、情報形態端末等、複数のフォーマットエンジンを動作させる情報処理装置に利用することができる。

請求の範囲

1．異なるフォーマットで記述されたデータをそれぞれ実行するためのフォーマットエンジンを複数格納している情報処理装置であって、

各フォーマットエンジンの動作状態を全てのフォーマットエンジンに共通の表現によって規定した共通状態を予め定義しておき、各フォーマットエンジンの動作を管理するフォーマットエンジン管理手段と、

各フォーマットエンジンに対応して設けられ、共通状態と、フォーマットエンジンの動作状態を各フォーマットエンジン毎に異なる表現によって規定した動作状態である個別状態との対応を予め定義しておき、任意の個別状態となるようにフォーマットエンジンの動作を制御する動作制御手段とを備え、

前記フォーマットエンジン管理手段は、あるフォーマットエンジンを所定の共通状態に変化させる場合、当該所定の共通状態を示す共通状態情報を含むメッセージを、当該フォーマットエンジンに対応して設けられた動作制御手段へ送信し、

前記動作制御手段は、前記フォーマットエンジン管理手段からメッセージが送信されてきた場合、当該メッセージに含まれる共通状態情報により示される共通状態に対応する個別状態となるように、当該フォーマットエンジンを制御する、情報処理装置。

2．各フォーマットエンジンに対応して設けられ、フォー

マットエンジンの個別状態と、当該個別状態に対応する共通状態との組によって構成されるテーブル格納するテーブル格納手段をさらに備え、

前記動作制御手段は、前記テーブルを参照することによって共通状態から個別状態を決定する、請求項１に記載の情報処理装置。

３．各フォーマットエンジンに対応して設けられ、フォーマットエンジンの個別状態を取得し、取得した個別状態に対応する共通状態を示す共通状態情報を前記フォーマットエンジン管理手段へ送信する個別状態取得手段をさらに備え、

前記フォーマットエンジン管理手段は、前記個別状態取得手段から出力された共通状態情報により示される共通状態に基づいて、各フォーマットエンジンの動作を管理する、請求項１に記載の情報処理装置。

４．フォーマットエンジンが実行中に利用するリソースであって、複数のフォーマットエンジンが同時に利用することが不可能なリソースである極小リソースをさらに備え、

前記個別状態取得手段は、フォーマットエンジンから取得した個別状態が極小リソースを利用している動作状態を示す場合、当該フォーマットエンジンの共通状態情報として、所定の状態を示す共通状態情報を前記フォーマットエンジン管理手段へ出力するとともに、フォーマットエンジンから取得した個別状態が極小リソースを利用していない動作状態を示す場合、当該フォーマットエンジンの共通状態情報として、前記所定の状態以外の状態を示す共通状態

情報を前記フォーマットエンジン管理手段へ出力し、

前記フォーマットエンジン管理手段は、共通状態情報が前記所定の状態を示すフォーマットエンジンが１つのみになるように、各フォーマットエンジンの動作を管理する、請求項３に記載の情報処理装置。

５．前記フォーマットエンジン管理手段は、

フォーマットエンジンを起動するための起動要求を受け付ける起動受付手段と、

前記起動受付手段が起動要求を受け付けたことに応じて、前記状態取得手段から各フォーマットエンジンの共通状態情報を取得する共通状態取得手段と、

前記共通状態取得手段によって取得された共通状態情報が実行中を示すフォーマットエンジンがある場合、当該フォーマットエンジンの動作を停止させるメッセージを当該フォーマットエンジンに対応して設けられた動作制御手段へ送信する動作停止手段と、

前記動作停止手段によってフォーマットエンジンの動作が停止された後、起動要求に対応するフォーマットエンジンを起動させるメッセージを当該フォーマットエンジンに対応して設けられた動作制御手段へ送信する起動手段とを含む、請求項４に記載の情報処理装置。

６．フォーマットエンジンが実行中に利用するリソースであって、複数のフォーマットエンジンが同時に利用することが不可能なリソースである極小リソースと、

フォーマットエンジンの要求に応じてフォーマットエンジンに対してリソースの利用を許可するリソース制御手段

と、

極小リソースを利用する場合における各フォーマットエンジン間の優先度を示す優先度情報を格納する優先度情報格納手段と、

極小リソースを利用する要求が複数のフォーマットエンジン間で重複する場合、前記優先度情報に基づいて、当該極小リソースの利用を許可すべきフォーマットエンジンを決定する許可決定手段とをさらに備え、

前記リソース制御手段は、極小リソースを利用する要求が複数のフォーマットエンジン間で重複する場合、前記許可決定手段によって決定されたフォーマットエンジンのみに当該極小リソースの利用を許可し、極小リソースを利用する要求が複数のフォーマットエンジン間で重複しない場合、当該要求を行ったフォーマットエンジンに当該極小リソースの利用を許可する、請求項1に記載の情報処理装置。

7．前記極小リソースは複数設けられ、

前記リソース制御手段は前記極小リソースに対応して複数設けられる、請求項6に記載の情報処理装置。

8．異なるフォーマットで記述されたデータをそれぞれ実行するためのフォーマットエンジンを格納している情報処理装置のコンピュータで実行可能なプログラムであって、

前記コンピュータを、

フォーマットエンジンの動作状態を全てのフォーマットエンジンに共通の表現によって規定した共通状態を予め定義しておき、各フォーマットエンジンの動作を管理する

フォーマットエンジン管理手段と、

各フォーマットエンジンに対応して設けられ、共通状態と、フォーマットエンジンの動作状態を各フォーマットエンジン毎に異なる表現によって規定した動作状態である個別状態との対応を予め定義しておき、任意の個別状態となるようにフォーマットエンジンの動作を制御する動作制御手段として機能させ、

前記フォーマットエンジン管理手段は、あるフォーマットエンジンを所定の共通状態に変化させる場合、当該所定の共通状態を示す共通状態情報を含むメッセージを、当該フォーマットエンジンに対応して設けられた動作制御手段へ送信し、

前記動作制御手段は、前記フォーマットエンジン管理手段からメッセージが送信されてきた場合、当該メッセージに含まれる共通状態情報により示される共通状態に対応する個別状態となるように、当該フォーマットエンジンを制御する、プログラム。

図 1

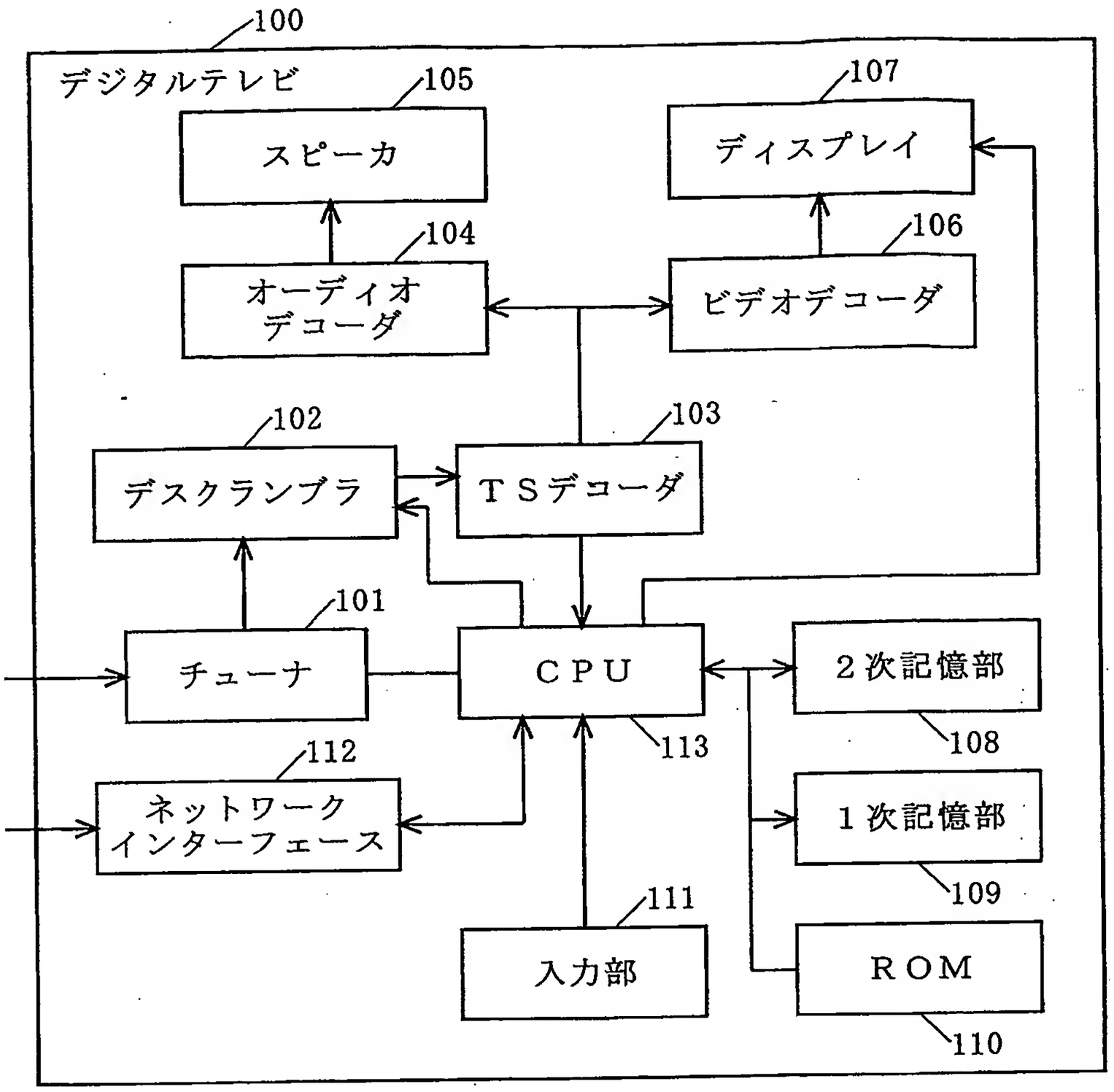

100
デジタルテレビ
105
スピーカ
107
ディスプレイ
104
オーディオ
デコーダ
106
ビデオデコーダ
102
デスクランブラ
103
ＴＳデコーダ
101
チューナ
ＣＰＵ
２次記憶部
113
108
112
ネットワーク
インターフェース
１次記憶部
109
111
入力部
ＲＯＭ
110

図 2

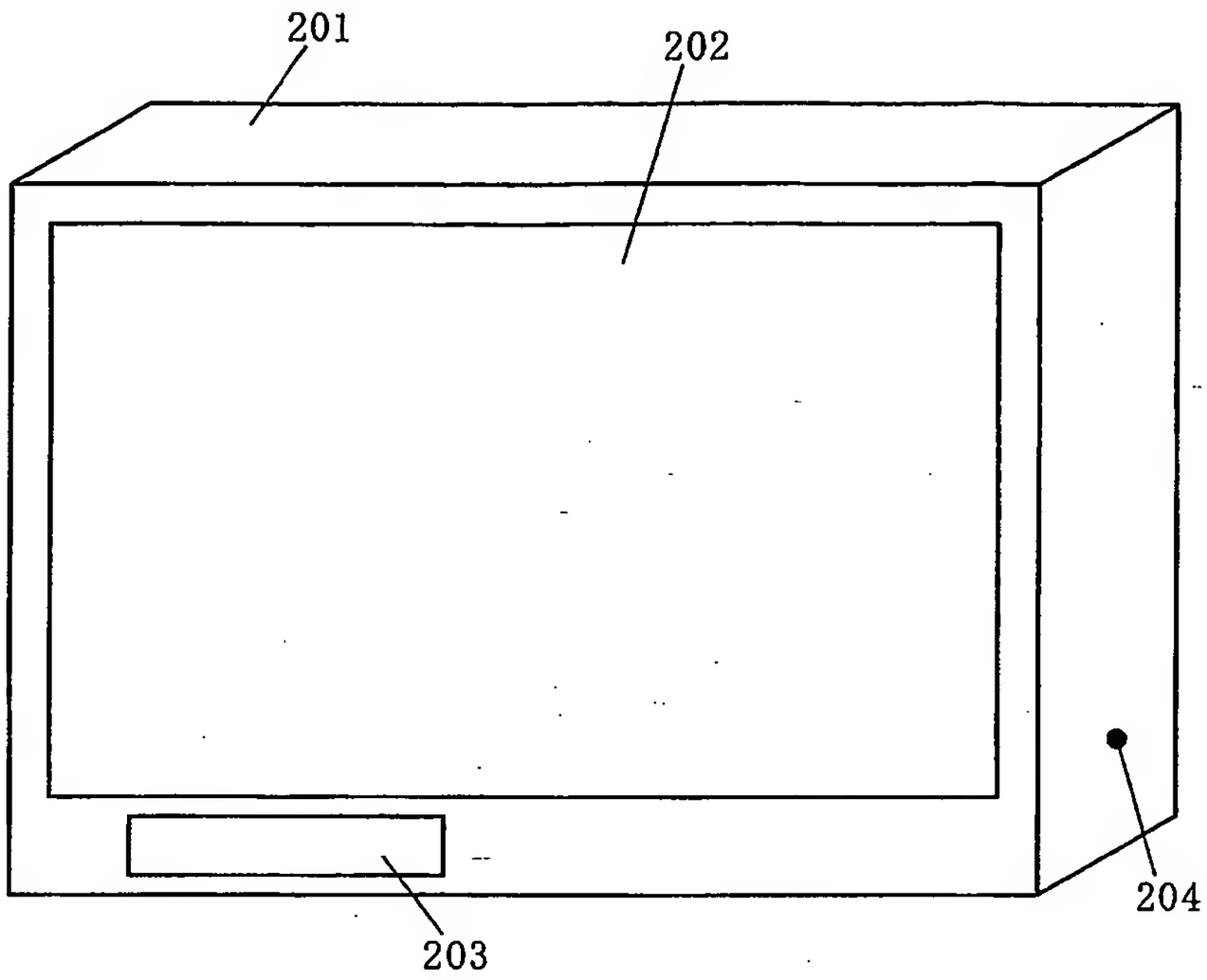

201
202
203
204

図 3

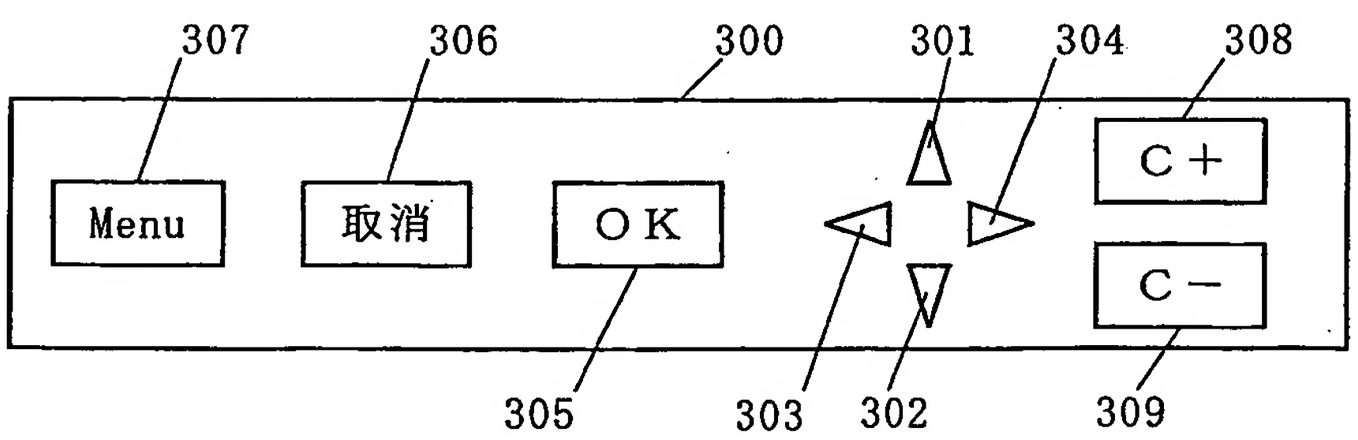

307
306
300
301
304
308
Menu
取消
OK
C+
C−
305
303
302
309

図 4

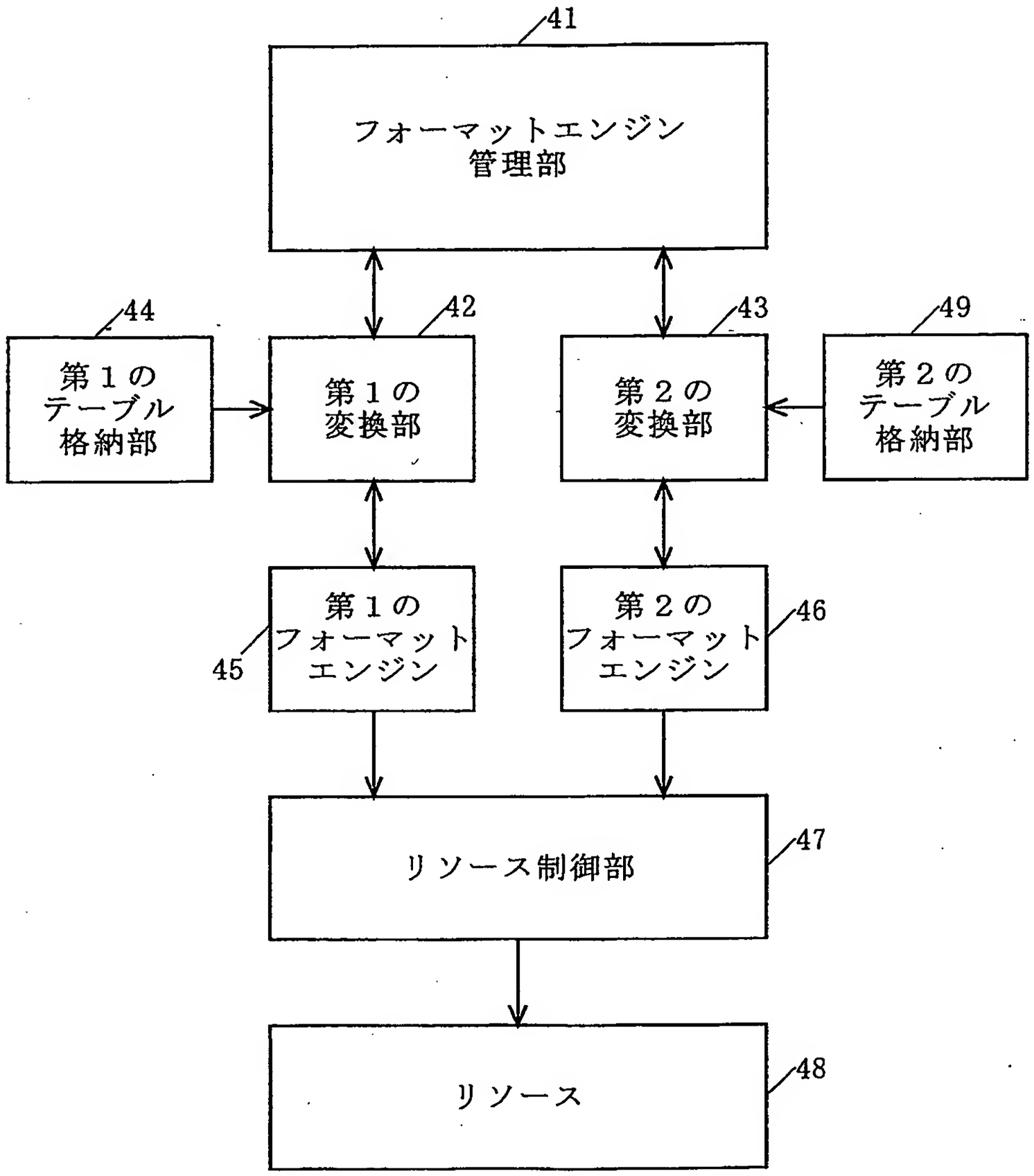

41
フォーマットエンジン
管理部
44
第１の
テーブル
格納部
42
第１の
変換部
43
第２の
変換部
49
第２の
テーブル
格納部
45
第１の
フォーマット
エンジン
46
第２の
フォーマット
エンジン
47
リソース制御部
48
リソース

図５

（ａ）

共通状態	第１のフォーマット エンジンの 個別状態
実行中	動作中
終了	停止中
	一時停止中

（ｂ）

共通状態	第２のフォーマット エンジンの 個別状態
実行中	起動中
終了	起動中でない

図 6

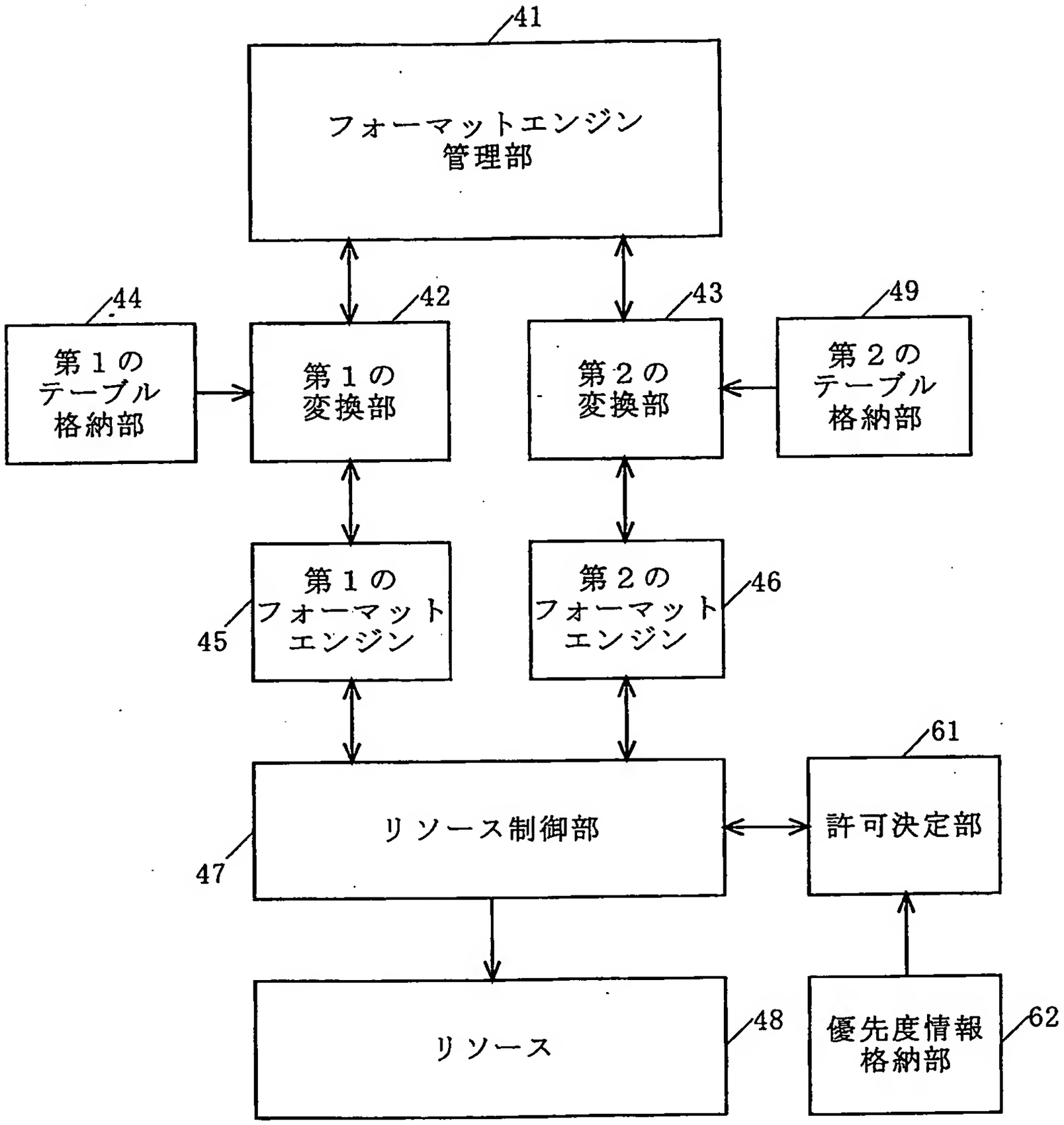

41
フォーマットエンジン
管理部
44
第１の
テーブル
格納部
42
第１の
変換部
43
第２の
変換部
49
第２の
テーブル
格納部
45
第１の
フォーマット
エンジン
46
第２の
フォーマット
エンジン
47
リソース制御部
61
許可決定部
48
リソース
62
優先度情報
格納部

図 7

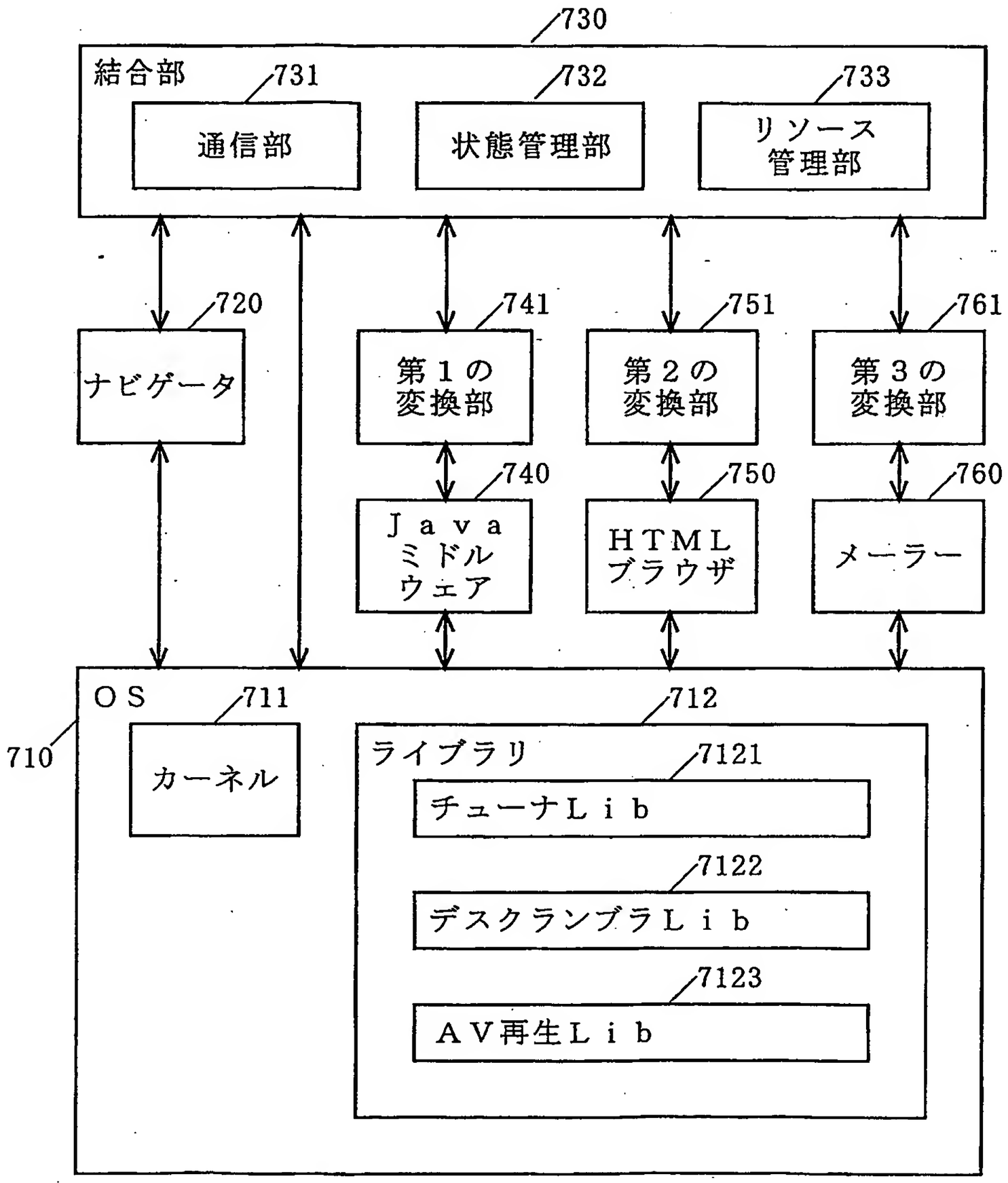

730
結合部
731
732
733
通信部
状態管理部
リソース
管理部
720
741
751
761
ナビゲータ
第１の
変換部
第２の
変換部
第３の
変換部
740
750
760
Ｊａｖａ
ミドル
ウェア
ＨＴＭＬ
ブラウザ
メーラー
ＯＳ
711
712
710
カーネル
ライブラリ
7121
チューナＬｉｂ
7122
デスクランブラＬｉｂ
7123
ＡＶ再生Ｌｉｂ

図８

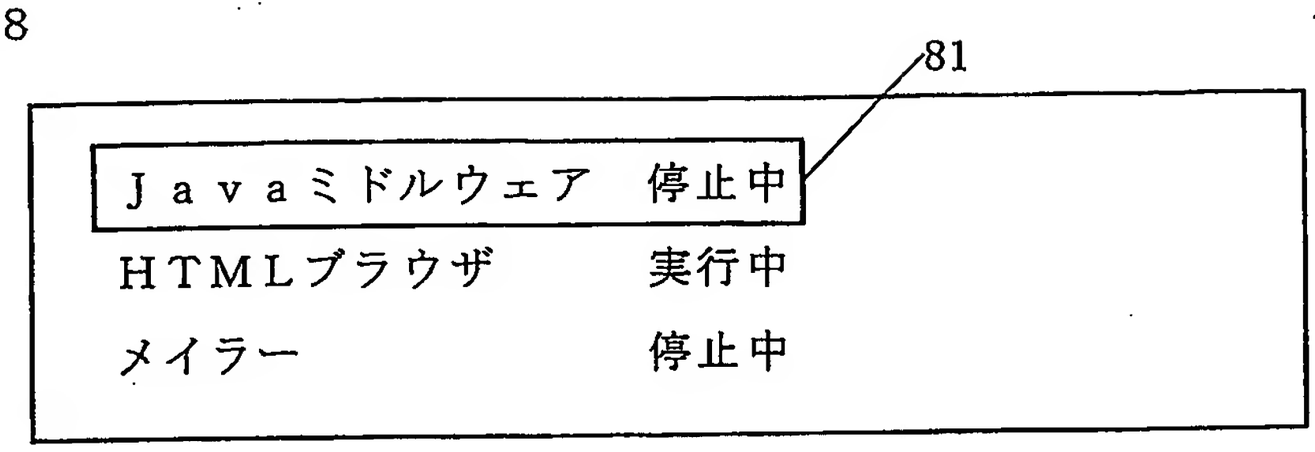

81
Ｊａｖａミドルウェア　停止中
ＨＴＭＬブラウザ　実行中
メイラー　停止中

図９

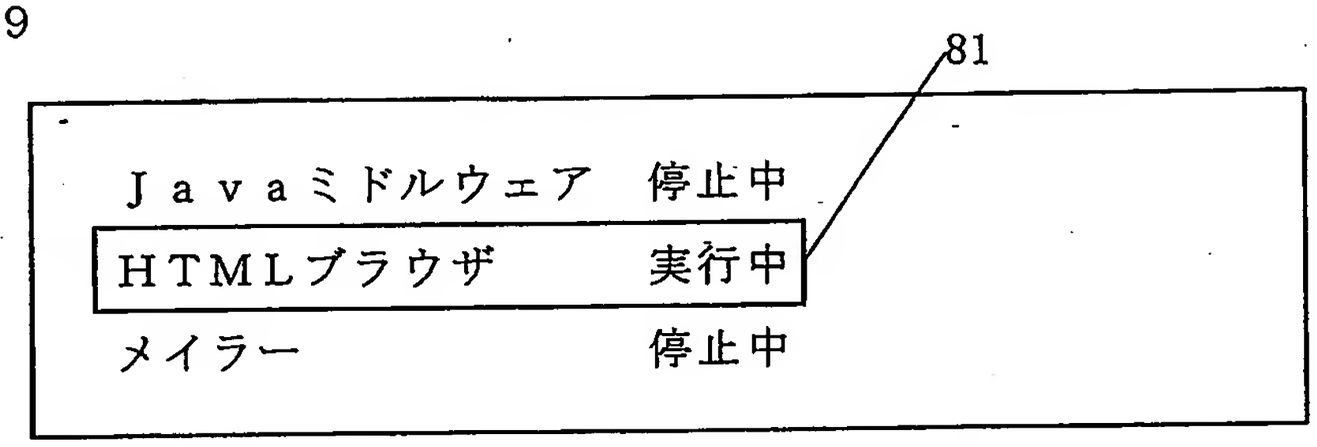

81
Ｊａｖａミドルウェア　停止中
ＨＴＭＬブラウザ　実行中
メイラー　停止中

図１０

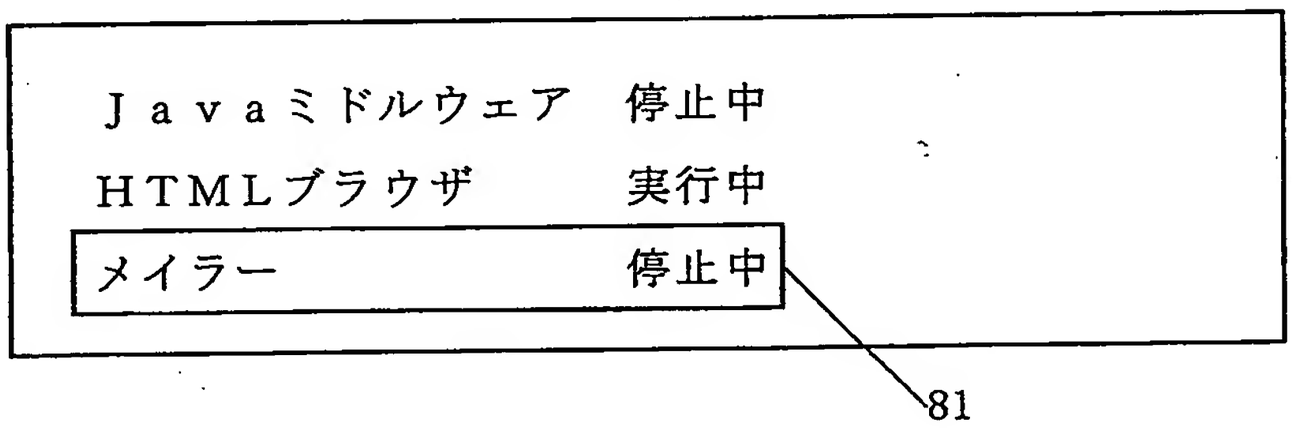

Ｊａｖａミドルウェア　停止中
ＨＴＭＬブラウザ　実行中
メイラー　停止中
81

図１１

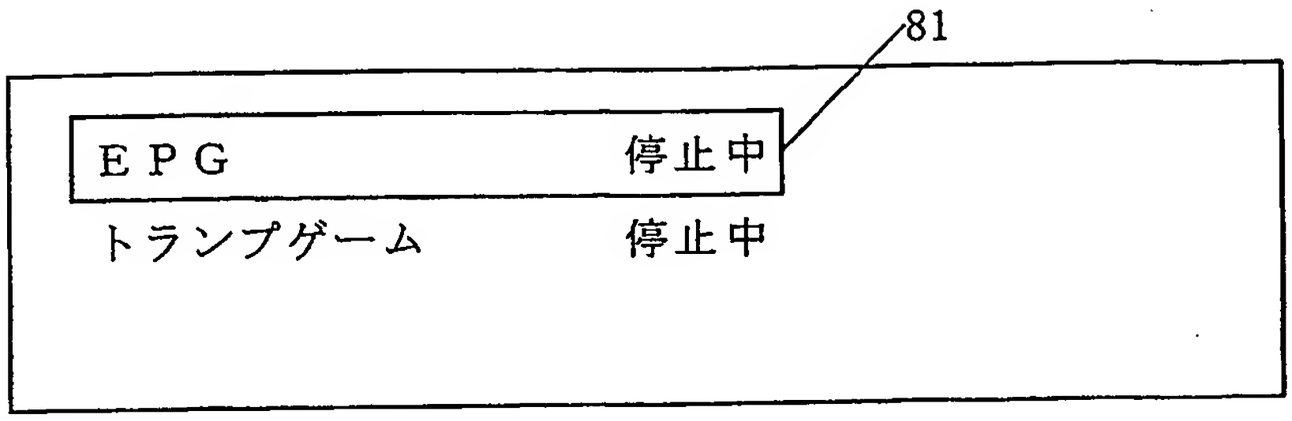


図１２

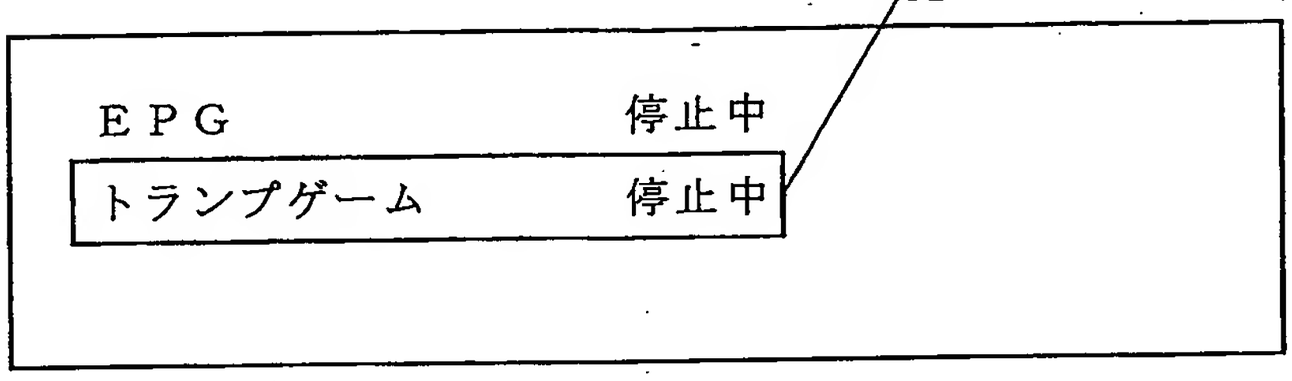


図１３

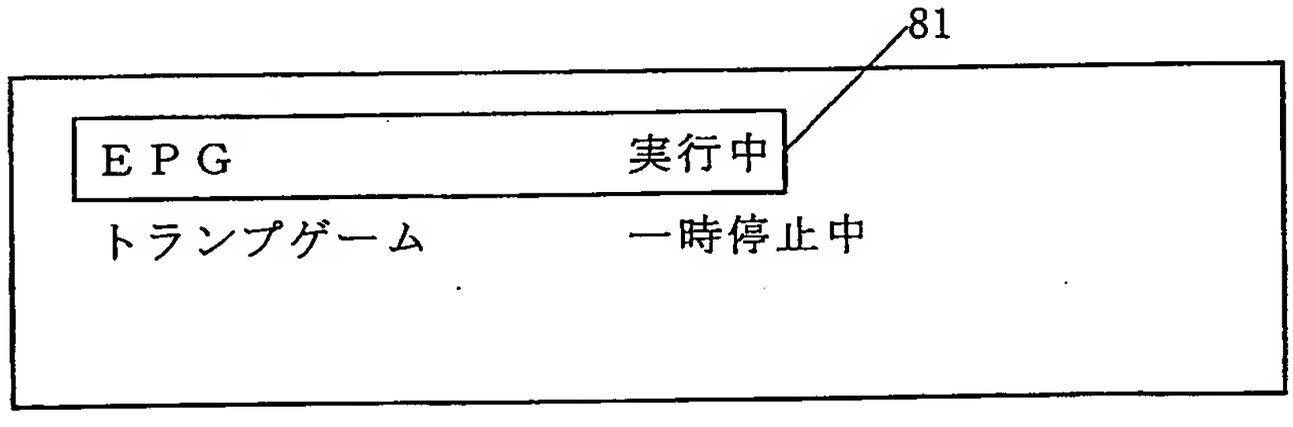


図１４

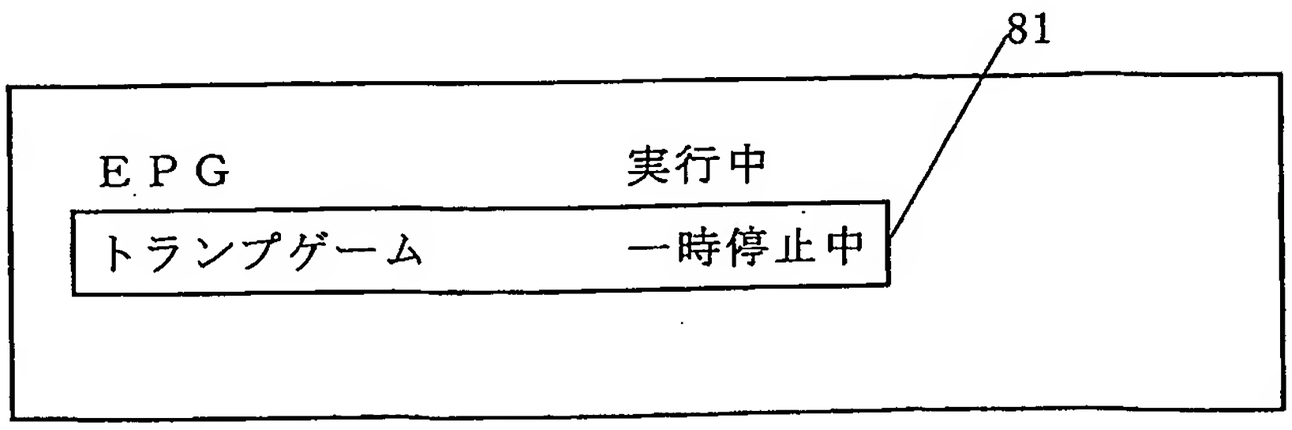


図１５

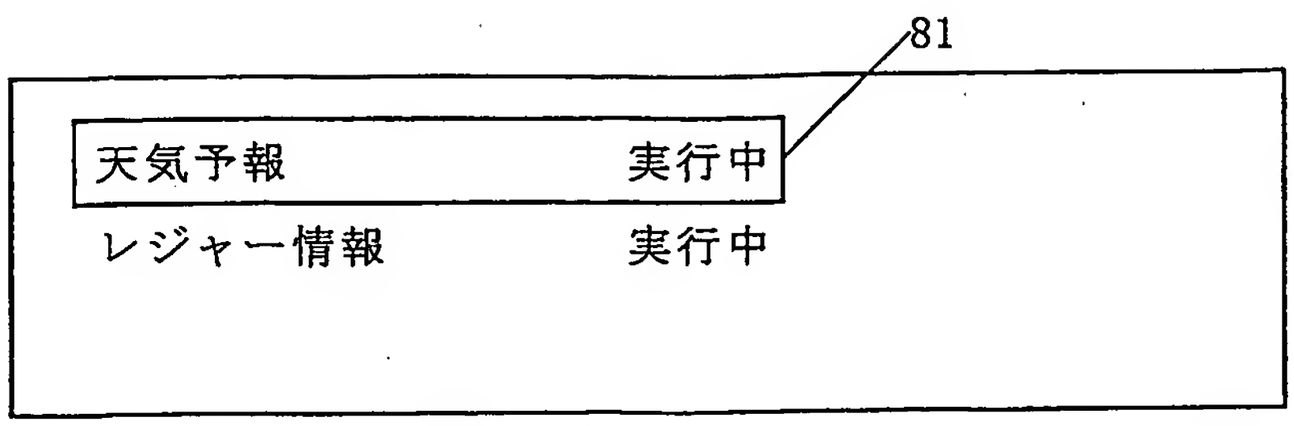
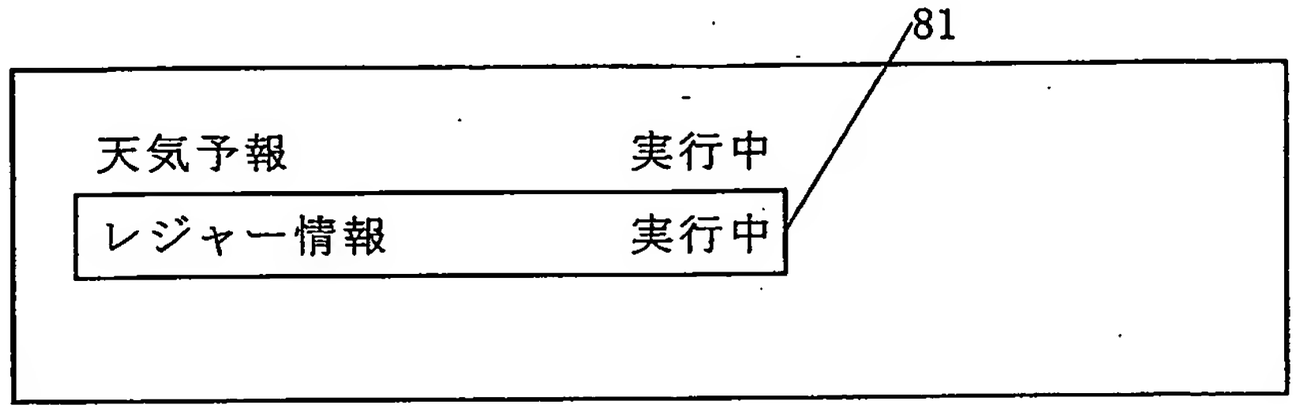


図１６



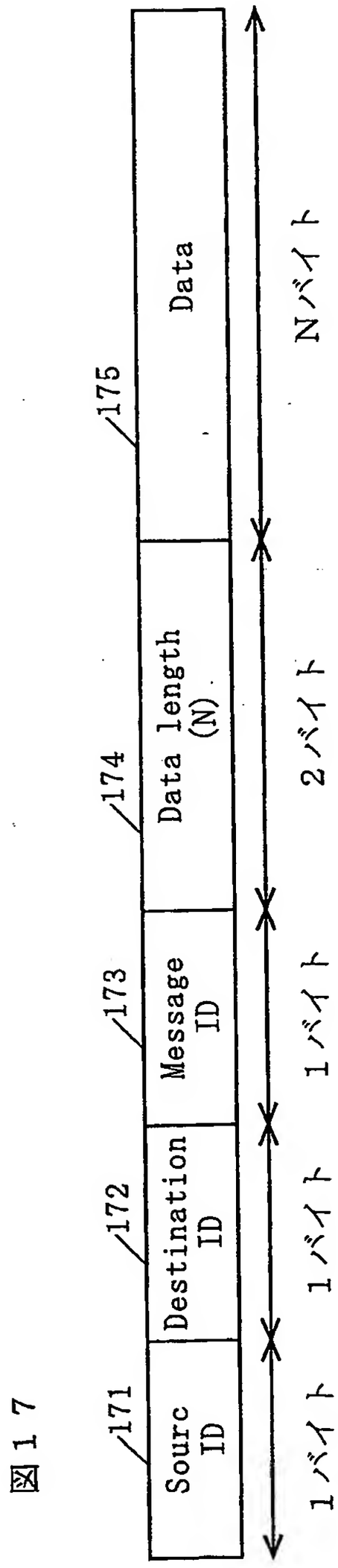

Source ID
Destination ID
Message ID
Data length (N)
Data
171
172
173
174
175
1バイト
1バイト
1バイト
2バイト
Nバイト

図17

図１８

サブプログラム	ＳＩＤ
ＡＬＬ	0
ナビゲータ	1
Ｊａｖａミドルウェア	2
ＨＴＭＬブラウザ	3
メイラー	4
結合部	5

図１９

メッセージ	ＭＩＤ	フォーマットＩＤ
フォーマットエンジン状態要求	1	なし
フォーマットエンジン状態応答	2	Ｆ０１
フォーマットエンジン状態変化	3	なし
フォーマットエンジン実行	4	なし
フォーマットエンジン終了	5	なし
フォーマットエンジン一時停止	6	なし
アプリケーション・データ一覧要求	１１	なし
アプリケーション・データ一覧応答	１２	Ｆ０２
アプリケーション・データ一覧変化	１３	なし
アプリケーション・データ実行	１４	Ｆ０３
アプリケーション・データ終了	１５	Ｆ０３
アプリケーション・データ一時停止	１６	Ｆ０３
プライベート	２１	－

図２０

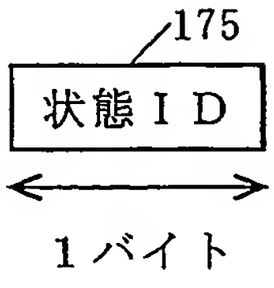


図２１

状態名	意味	動作状態ＩＤ
実行中	極小リソースを使用中	1
一時停止中	共有可能リソースを使用中	2
停止中	リソース未使用	3

図２２

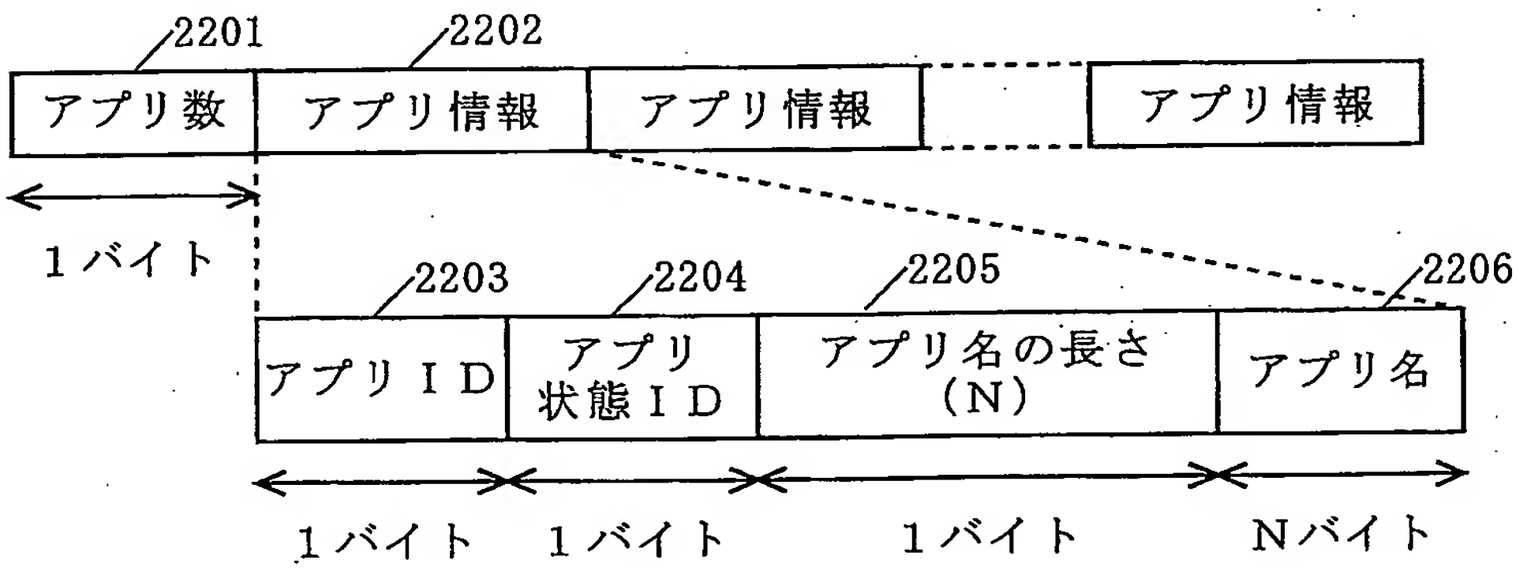


図２３

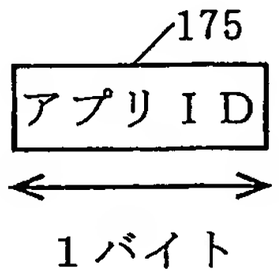


図２４

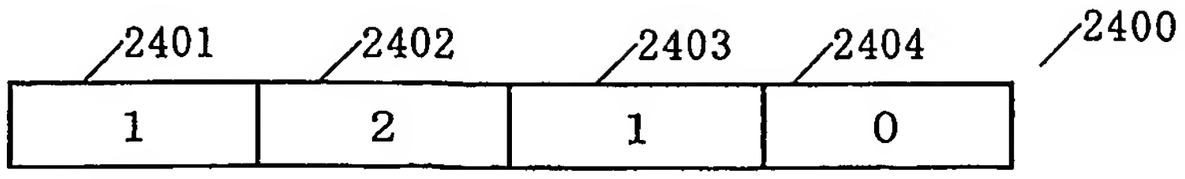


図２５

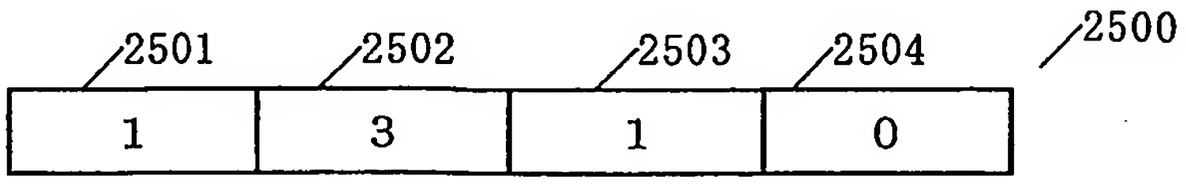


図２６

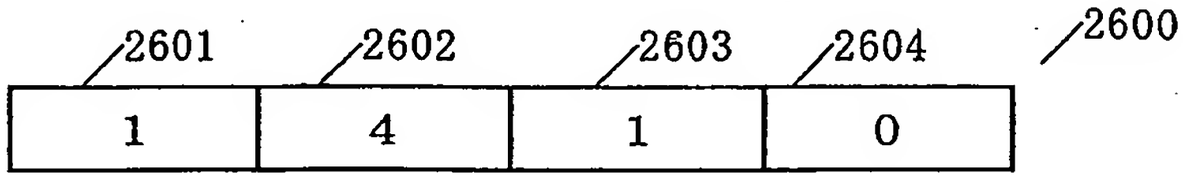


図２７

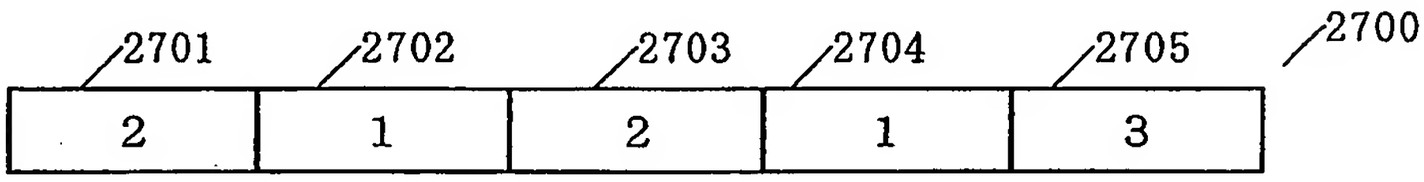


図２８

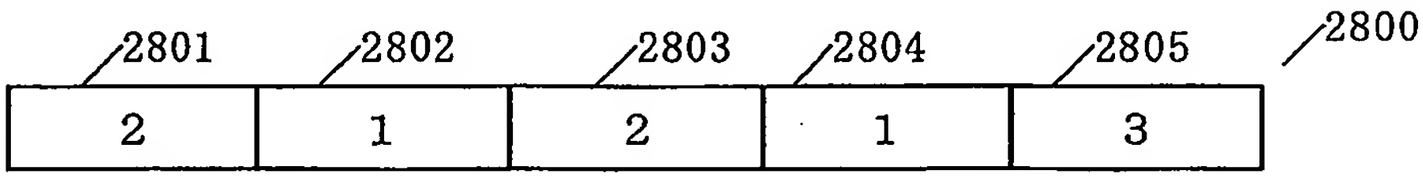


図２９

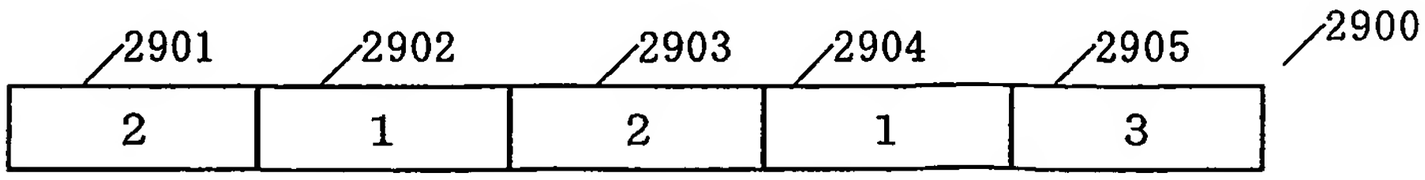


図３０

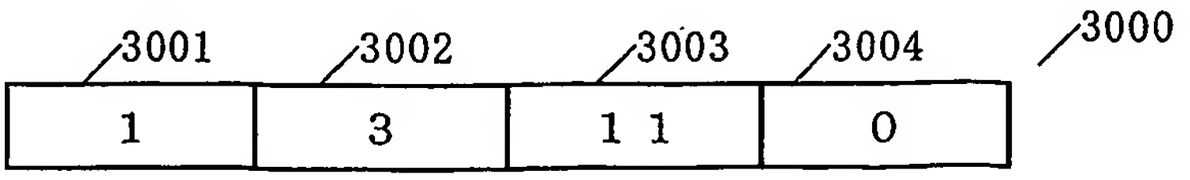


図３１

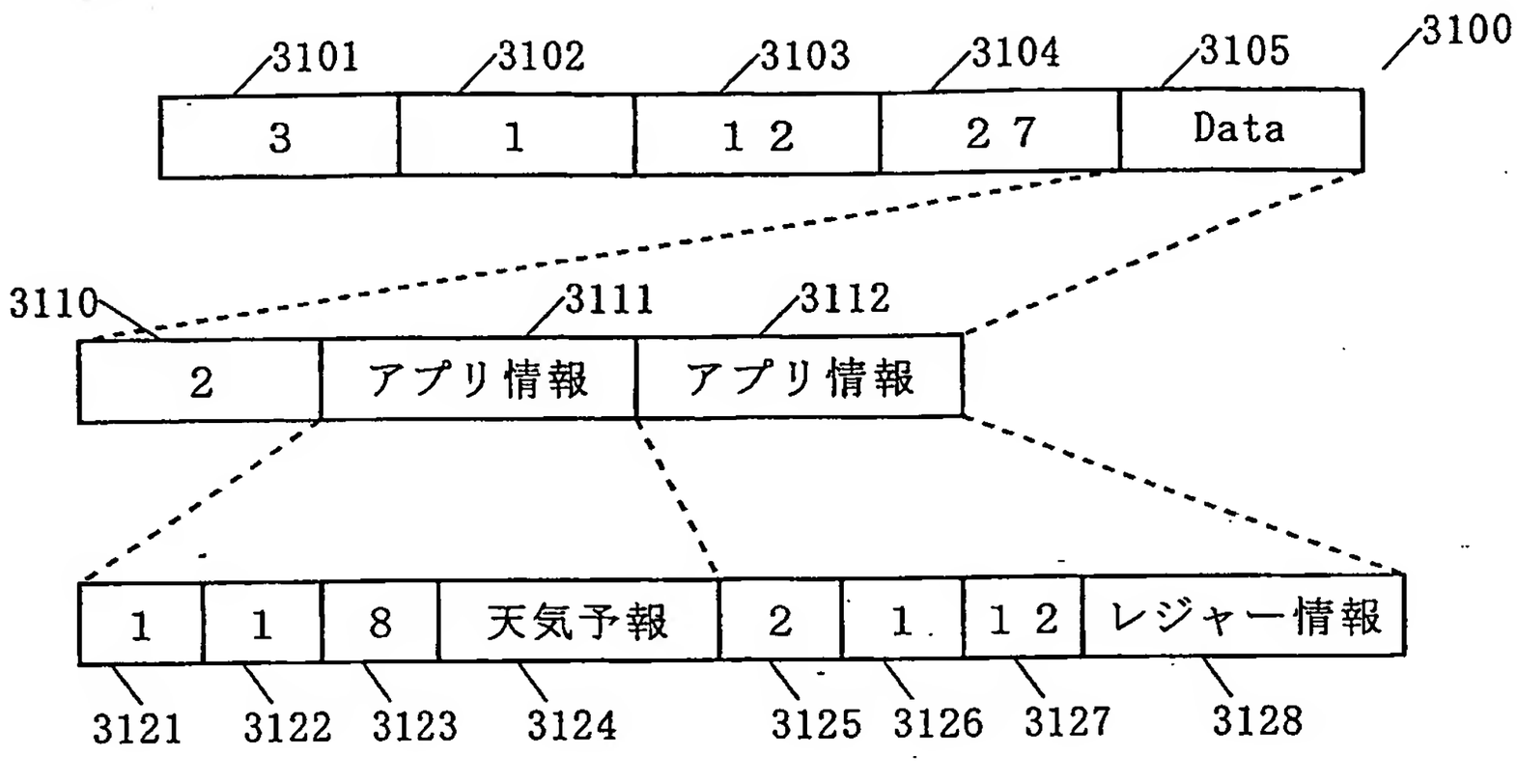

3100
3101
3102
3103
3104
3105
3
1
１２
２７
Data
3110
3111
3112
2
アプリ情報
アプリ情報
1
1
8
天気予報
2
1
１２
レジャー情報
3121
3122
3123
3124
3125
3126
3127
3128

図３２

ＦＩＤ	フォーマットエンジン名	状態ＩＤ	アプリ情報
2	Ｊａｖａミドルウェア	3	0x5678
3	ＨＴＭＬブラウザ	3	0x7162
4	メイラー	3	NULL

図３３

ＡＩＤ	アプリ名	状態ＩＤ
1	ＥＰＧ	3
2	トランプゲーム	3

図３４

データＩＤ	アプリ名	状態ＩＤ
1	天気予報	1
2	レジャー情報	1

図３５

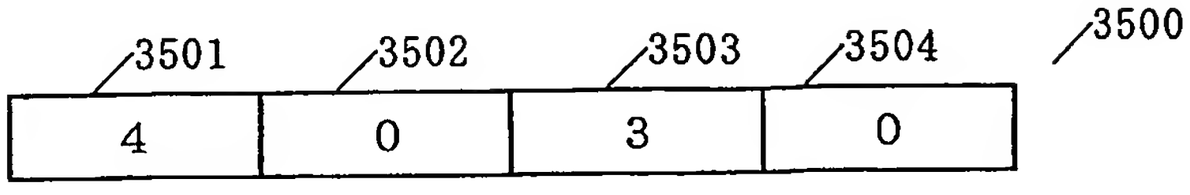


図３６

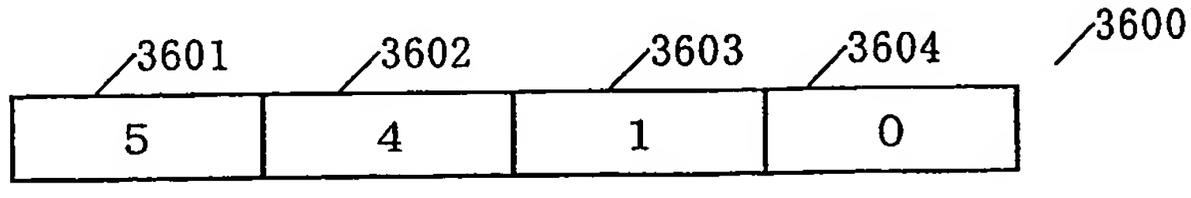


図３７

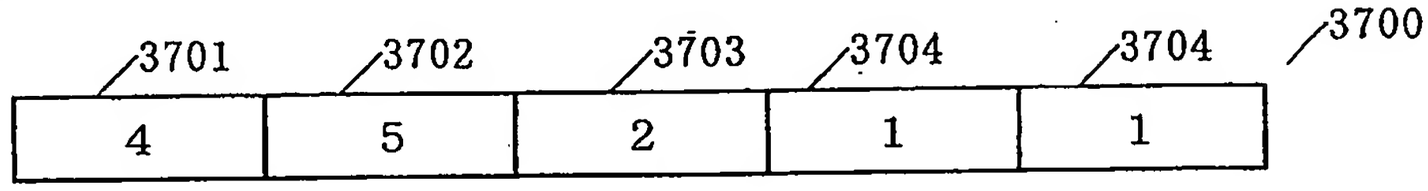


図３８

ＦＩＤ	フォーマットエンジン名	状態ＩＤ	アプリ情報
2	Ｊａｖａミドルウェア	3	0x5678
3	ＨＴＭＬブラウザ	3	0x7162
4	メイラー	1	NULL

図３９

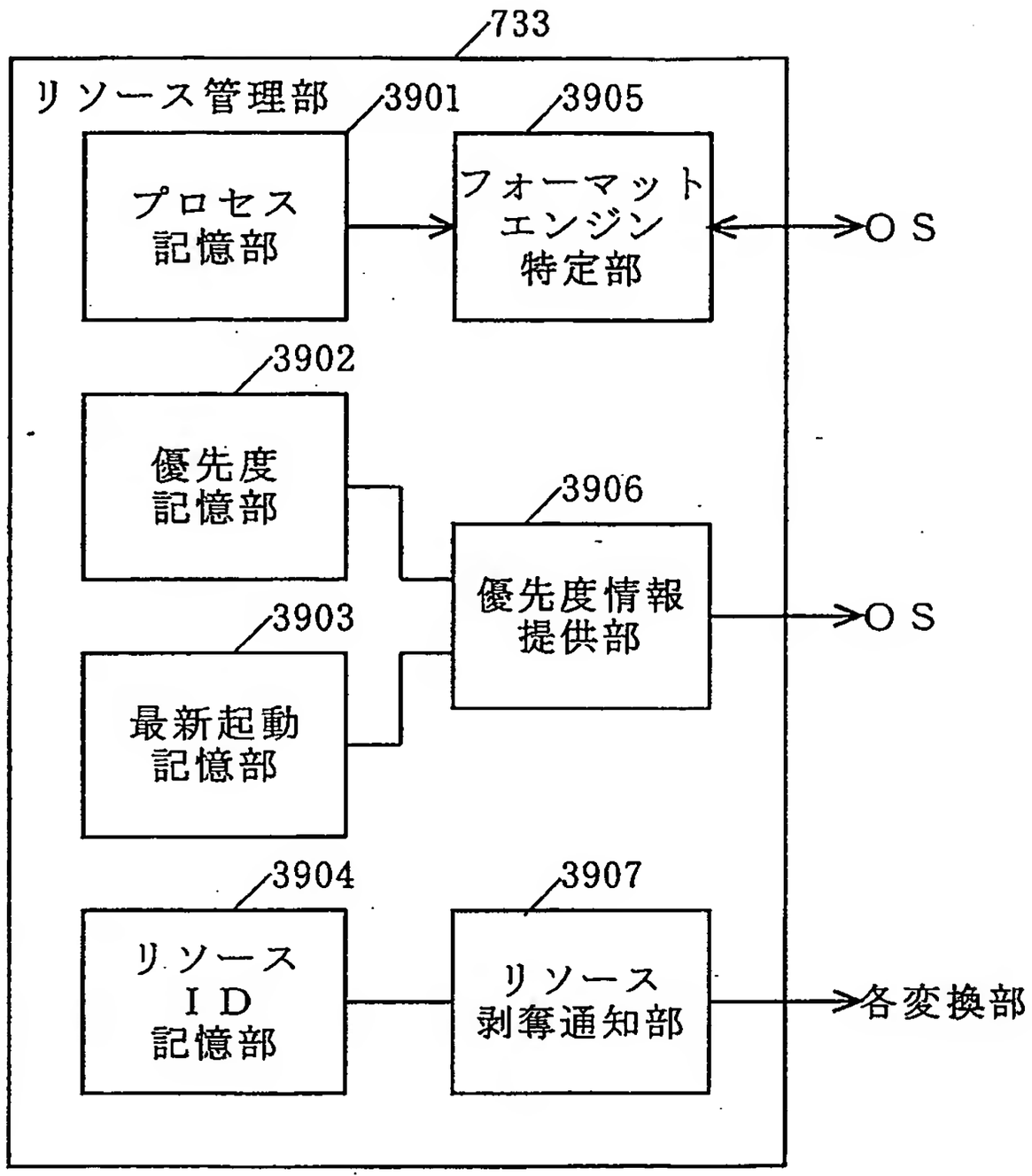

733
リソース管理部
3901
3905
プロセス記憶部
フォーマットエンジン特定部
ＯＳ
3902
優先度記憶部
3906
優先度情報提供部
ＯＳ
3903
最新起動記憶部
3904
3907
リソースＩＤ記憶部
リソース剥奪通知部
各変換部

図４０

ＦＩＤ	プロセスＩＤ
2	１００
3	１１０
4	１２０

図４１

ＦＩＤ	優先度
2	2
3	1
4	3

図４２

4

図４３

ＲＩＤ	リソース
1	チューナ
2	ＴＳデコーダ
3	オーディオデコーダ
4	ビデオデコーダ
5	ネットワークインターフェース

図４４

```
#define RESOURCE_ID_TUNER            1
#define RESOURCE_ID_TSDECODER        2
#define RESOURCE_ID_AUDIODECODER     3
#define RESOURCE_ID_VIDEODECODER     4
#define RESOURCE_ID_NETWORK          5
```

図４５

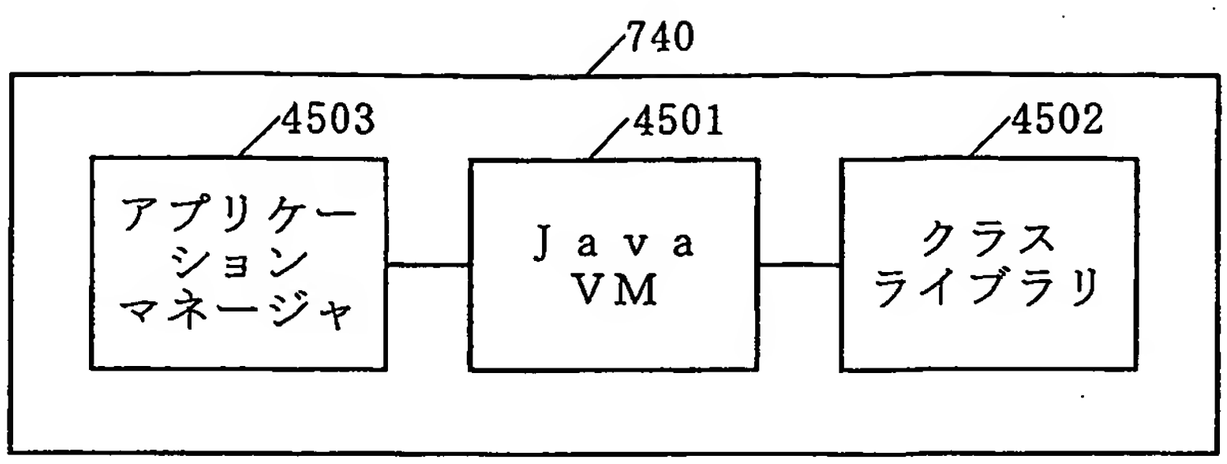


図４６

ＡＩＤ	制御情報	アプリ名	アプリ優先度
1	autostart	ＥＰＧ	６４
2	present	トランプゲーム	３２

図４７

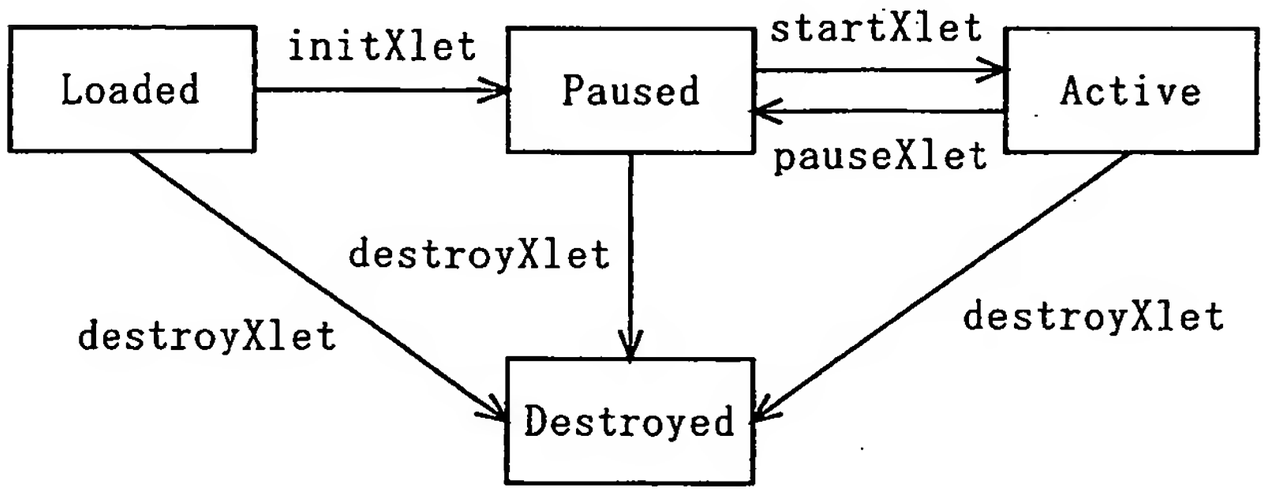


図４８

ＡＩＤ	制御情報	アプリ名	アプリ優先度
1	kill	ＥＰＧ	６４
2	present	トランプゲーム	３２

図４９

アプリケーションの状態	Ｊａｖａミドルウェアの状態
少なくとも１つの アプリケーションの状態が 「Ａｃｔｉｖｅ」	実行中
「Ａｃｔｉｖｅ」状態の アプリケーションがなく、かつ、 少なくとも１つの アプリケーションの状態が 「Ｐａｕｓｅｄ」	一時停止中
上記以外	停止中

図５０

アプリケーションの状態	結合部が規定する状態
Ａｃｔｉｖｅ	実行中
Ｐａｕｓｅｄ	一時停止中
上記以外	停止中

図５１

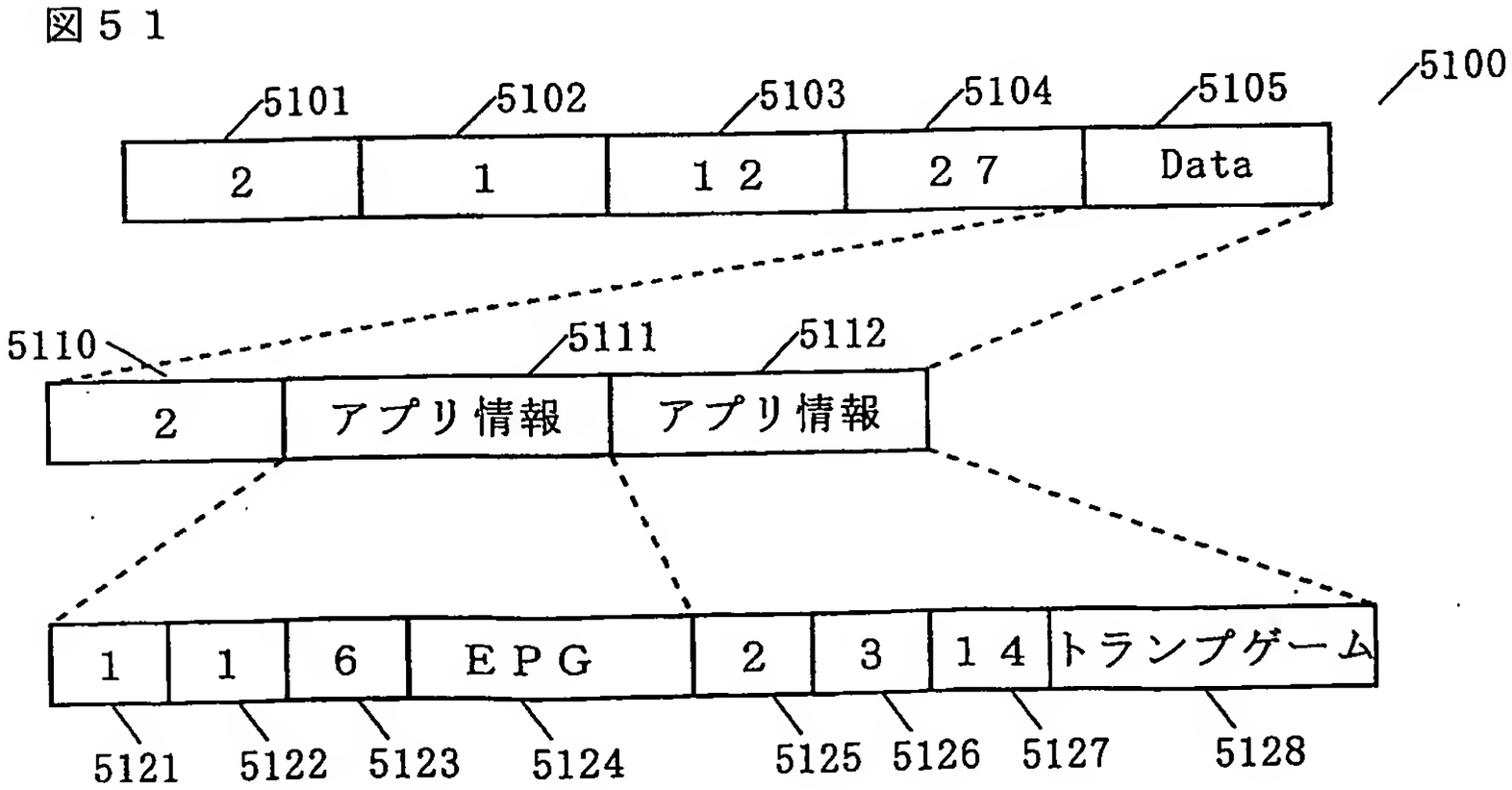


図５２

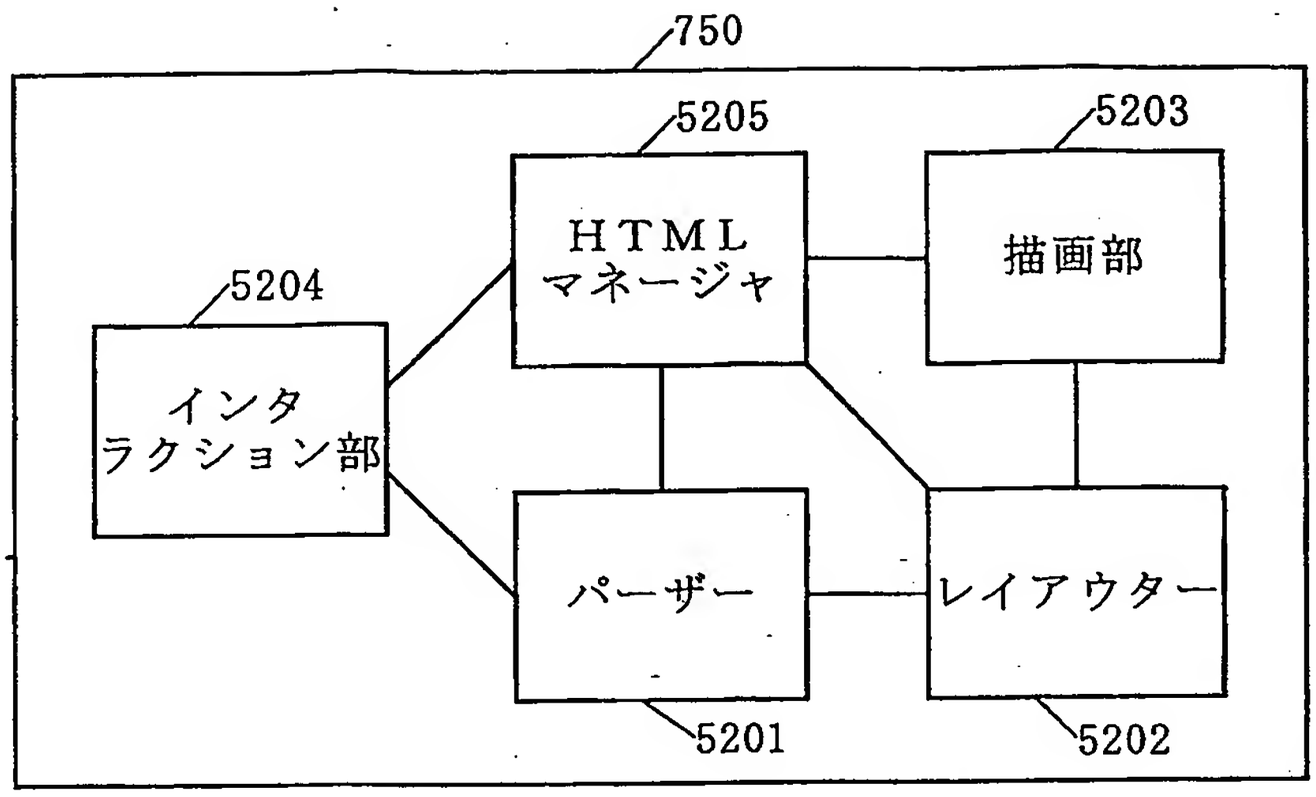

750
5205
HTML
マネージャ
5203
描画部
5204
インタ
ラクション部
パーザー
レイアウター
5201
5202

図５３

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//DVB//DTD XHTML DVB-
HTML 1.0//EN"
"http://www.dvb.org/mhp/dtd/dvbhtml-1-0.dtd">
<html xmlns="http://www.w3c.org/1999/xhtml">
<head>
<link rel="stylesheet" href="sampleCSS.css" type="text/css">
</head>
<body  id="body">                                          5311
<H1>各国のWeather forecast サービス紹介</H1>
<a class="param1" href="dvb://1.2.1/Japan_Weather.html">
Japan</a>                                                  5312
<br></br>
<a class="param2" href="dvb://1.2.2/USA_Weather.html">
USA</a>                                                    5313
<br></br>
<a class="param3"                                          5314
href="dvb://1.2.3/Germany_Weather.html">
Germany</a>
</body>
</html>
```

図５４

```
H1 {position:absolute;top:50px;left:50px;}
a.param1{position:absolute;top:200px;left:100px;}
a.param2{position:absolute;top:300px;left:100px;}
a.param3{position:absolute;top:400px;left:100px;}
```

図５５

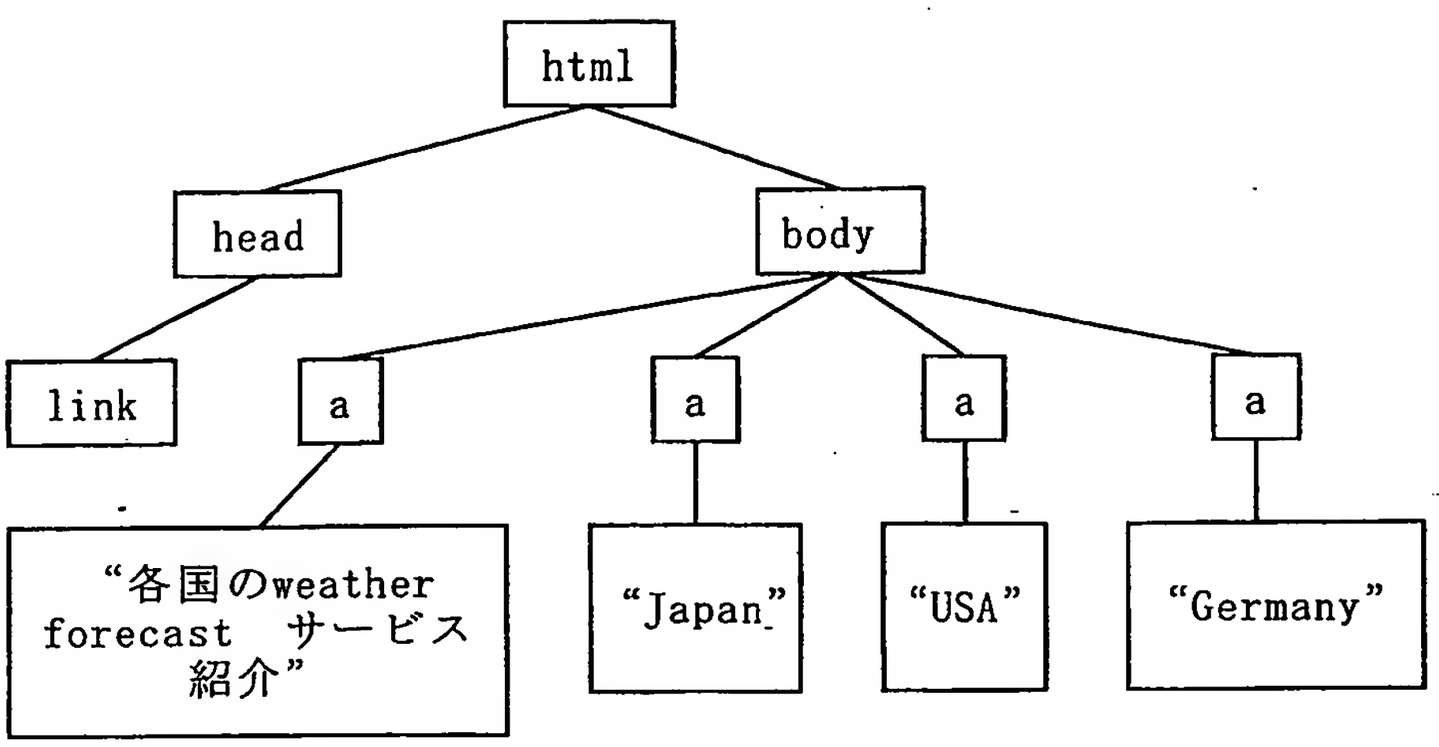

html
head
body
link
a
a
a
a
“各国のweather forecast　サービス紹介”
“Japan”
“USA”
“Germany”

図５６

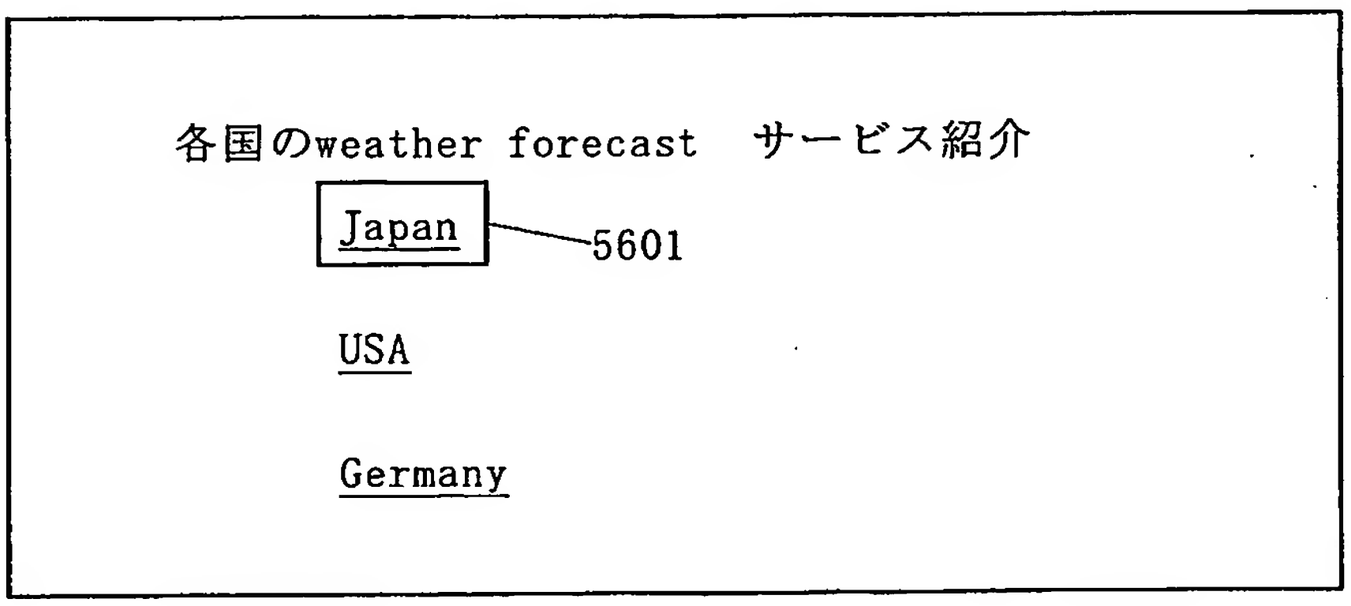

各国のweather forecast　サービス紹介
Japan
5601
USA
Germany

図５７

天気予報

東京　　くもり

大阪　　はれ

福岡　　雨

図５８

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//DVB//DTD XHTML DVB-HTML
1.0//EN"
"http://www.dvb.org/mhp/dtd/dvbhtml-1-0.dtd">
<html xmlns="http://www.w3c.org/1999/xhtml">
<head>
<script type="text/javascript">                    5801
 <!--
  function ChangeDOMTree(){
  ...
  }
//-->
 </script>
</head>
<body>
 <table>
  <tr>
     <td>海 </td>
  </tr>
  <tr>
     <td>山</td>
  </tr>
 </table>                                          5802
 <form>
  <input type="button" value="詳細情報"
   onclick= ChangeDOMTree() >
  </input>
 </form>
</body>
<html>
```

図５９

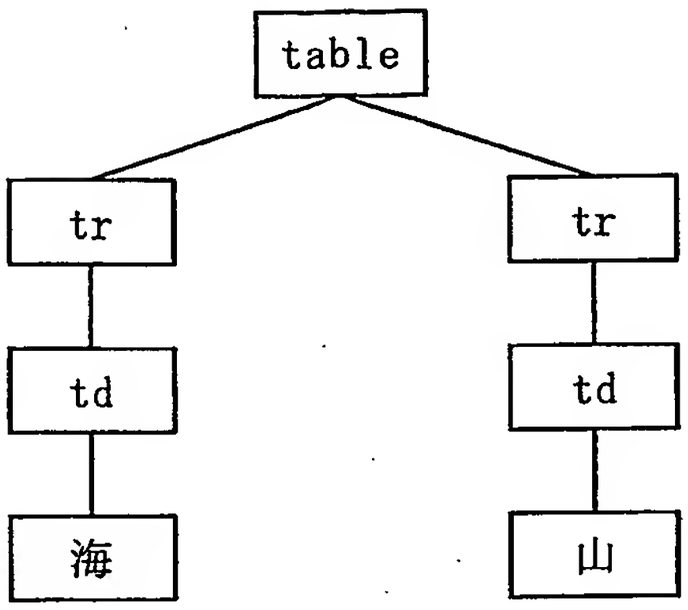


図６０

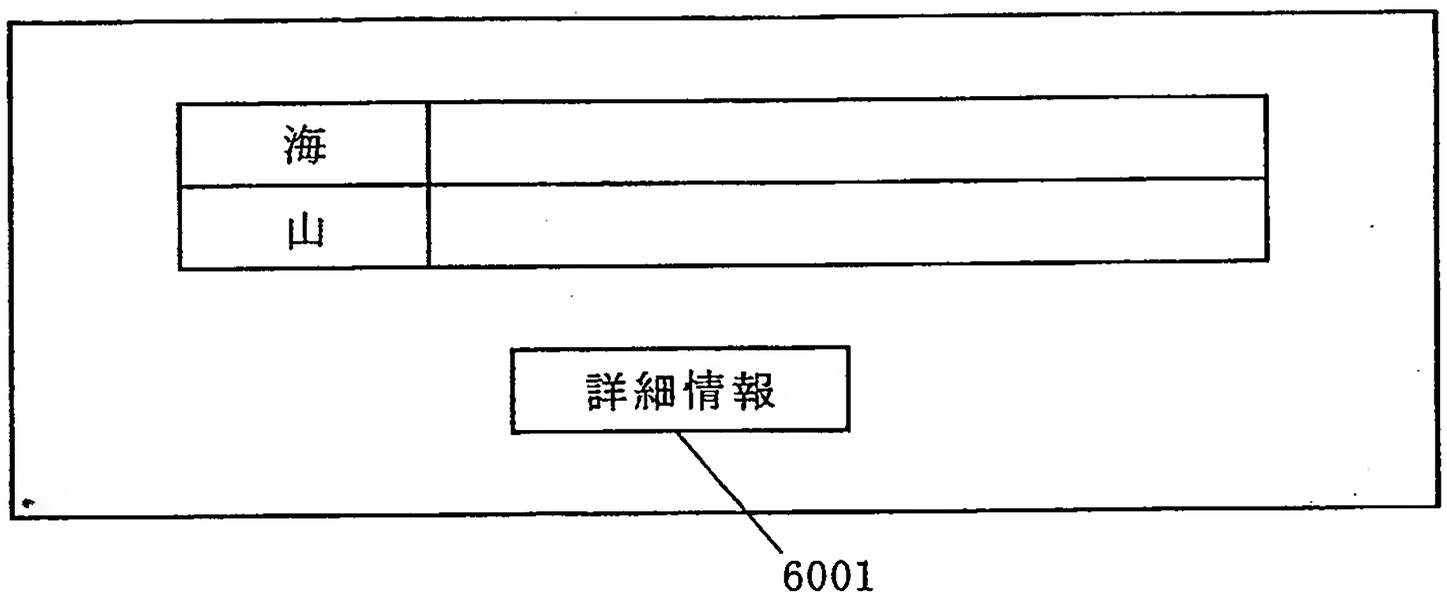


図６１

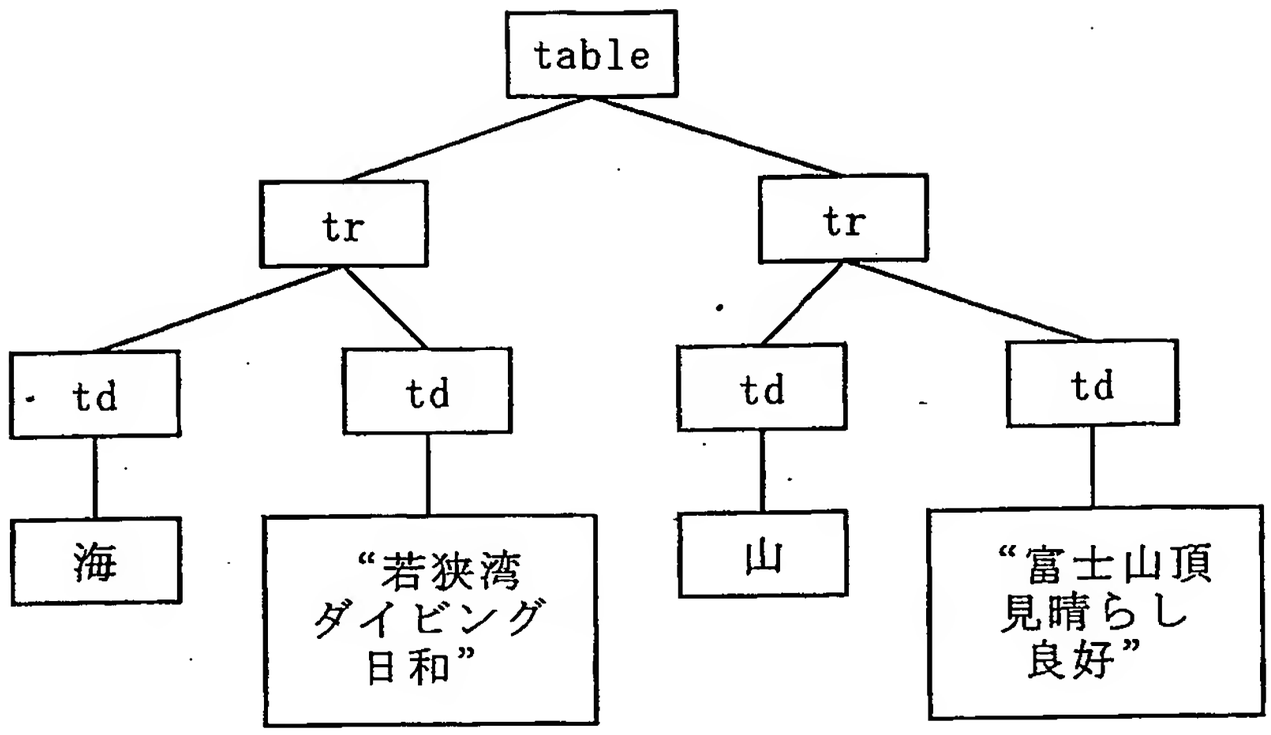


table
tr
tr
td
td
td
td
海
“若狭湾
ダイビング
日和”
山
“富士山頂
見晴らし
良好”

図６２

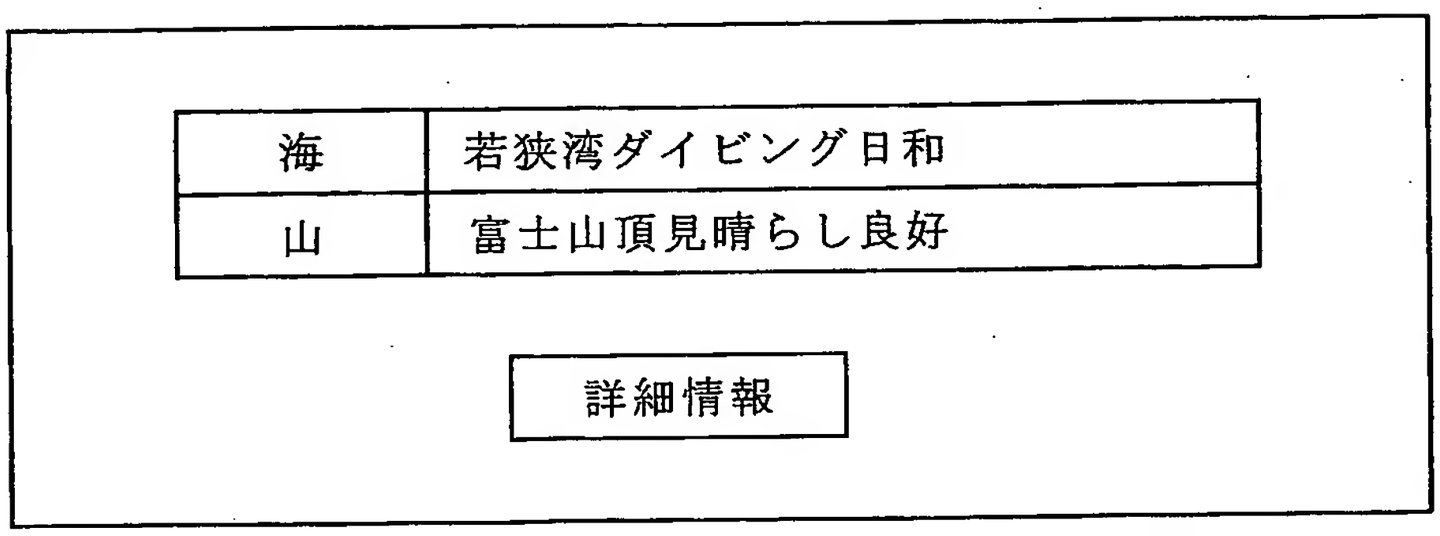

海
若狭湾ダイビング日和
山
富士山頂見晴らし良好
詳細情報

図６３

データＩＤ	制御情報	データ名	アプリ優先度
1	autostart	天気予報	５４
2	present	レジャー情報	２２

図６４

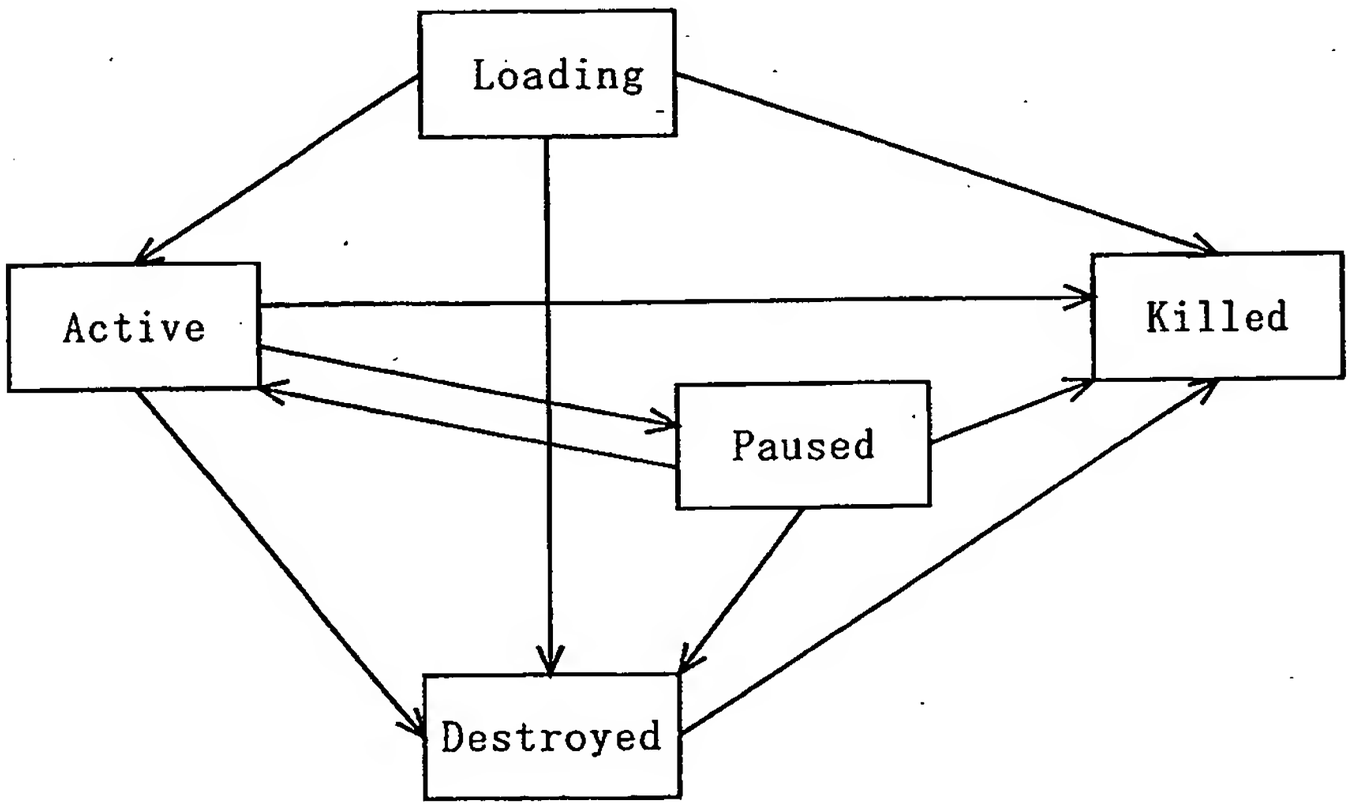


図６５

データＩＤ	制御情報	データ名	アプリ優先度
1	kill	天気予報	５４
2	present	レジャー情報	２２

図６６

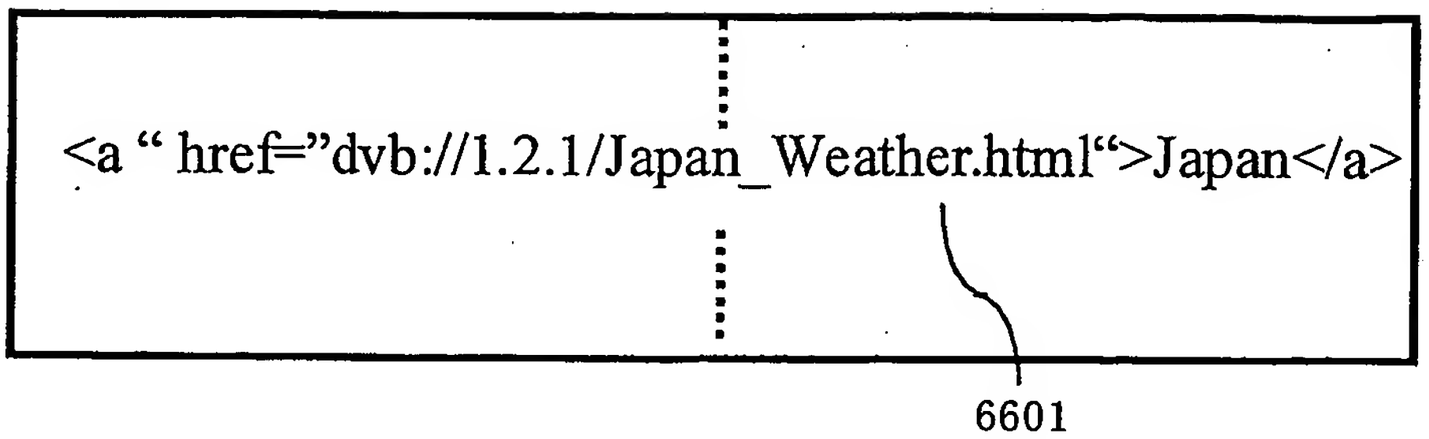


図６７

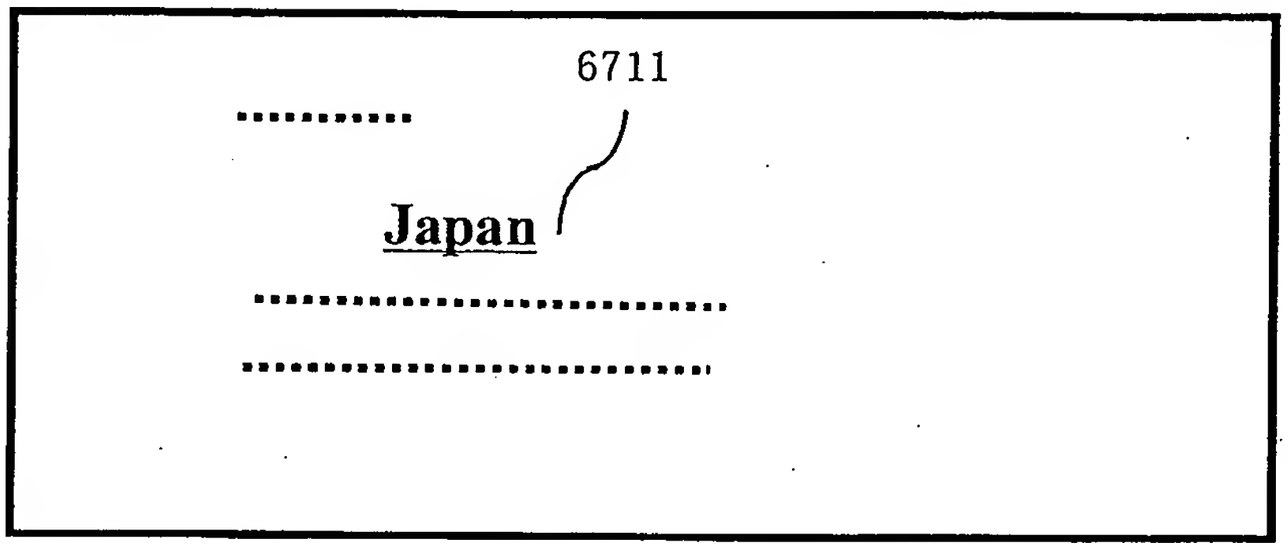


図６８

天気予報

東京　　くもり
大阪　　はれ
福岡　　雨

図６９

ＨＴＭＬデータの状態	ＨＴＭＬブラウザの状態
少なくとも１つの ＨＴＭＬデータの状態が 「Ａｃｔｉｖｅ」	実行中
「Ａｃｔｉｖｅ」状態の ＨＴＭＬデータがなく、かつ、 少なくとも１つの ＨＴＭＬデータの状態が 「Ｐａｕｓｅｄ」	一時停止中
上記以外	停止中

図７０

ＨＴＭＬデータの状態	結合部が規定する状態
Ａｃｔｉｖｅ	実行中
Ｐａｕｓｅｄ	一時停止中
上記以外	停止中

図７１

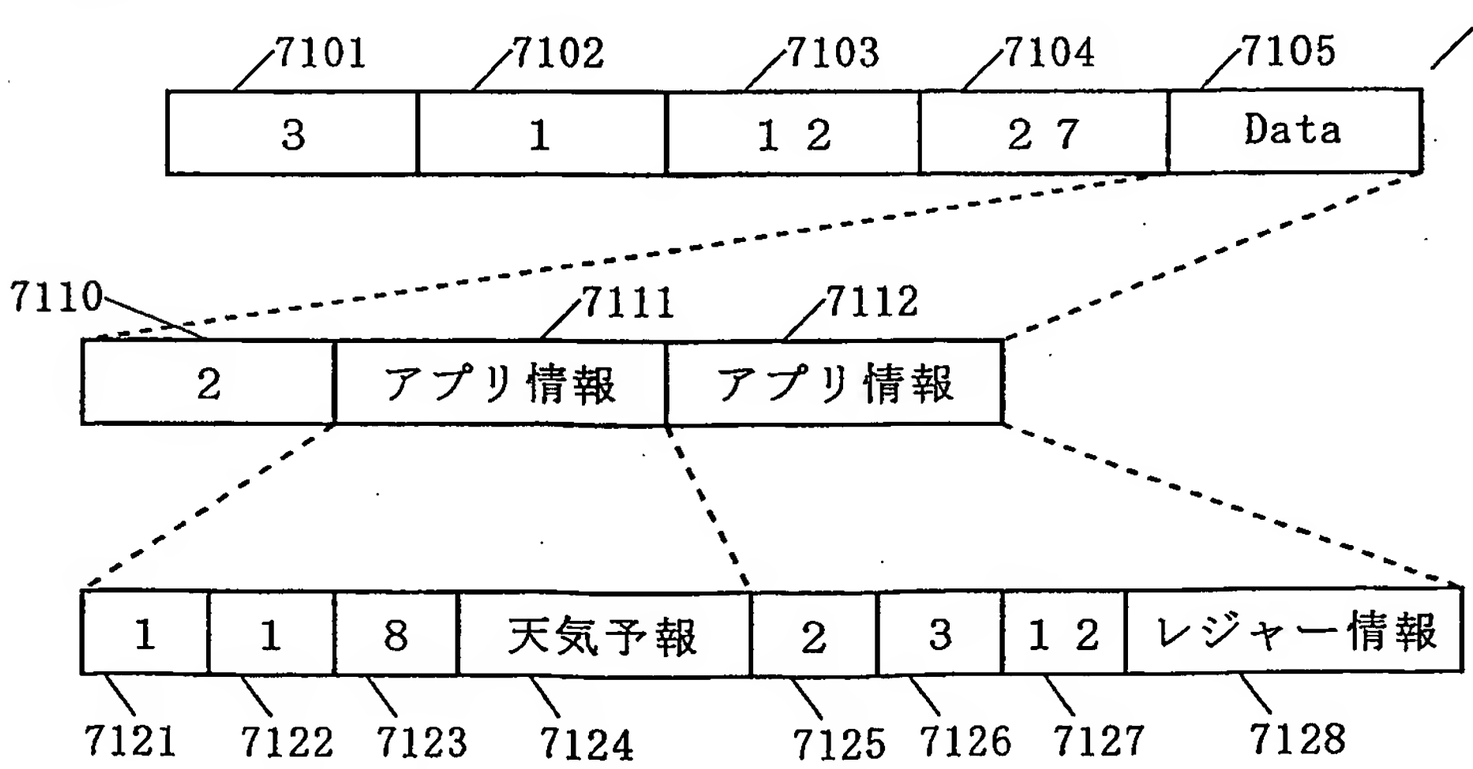


図７２

メイラーの状態	結合部が規定する状態
起動中	実行中
停止中	停止中

図７３

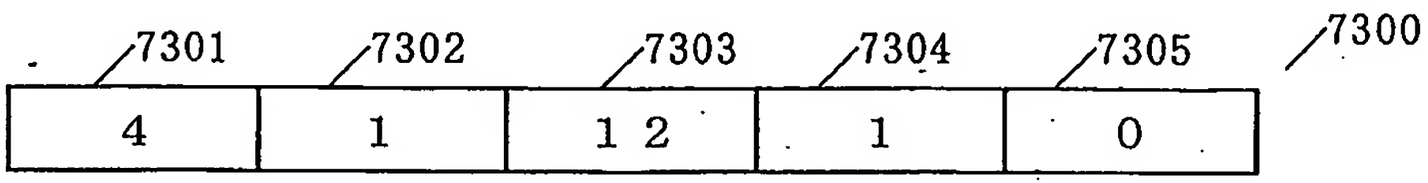


図７４

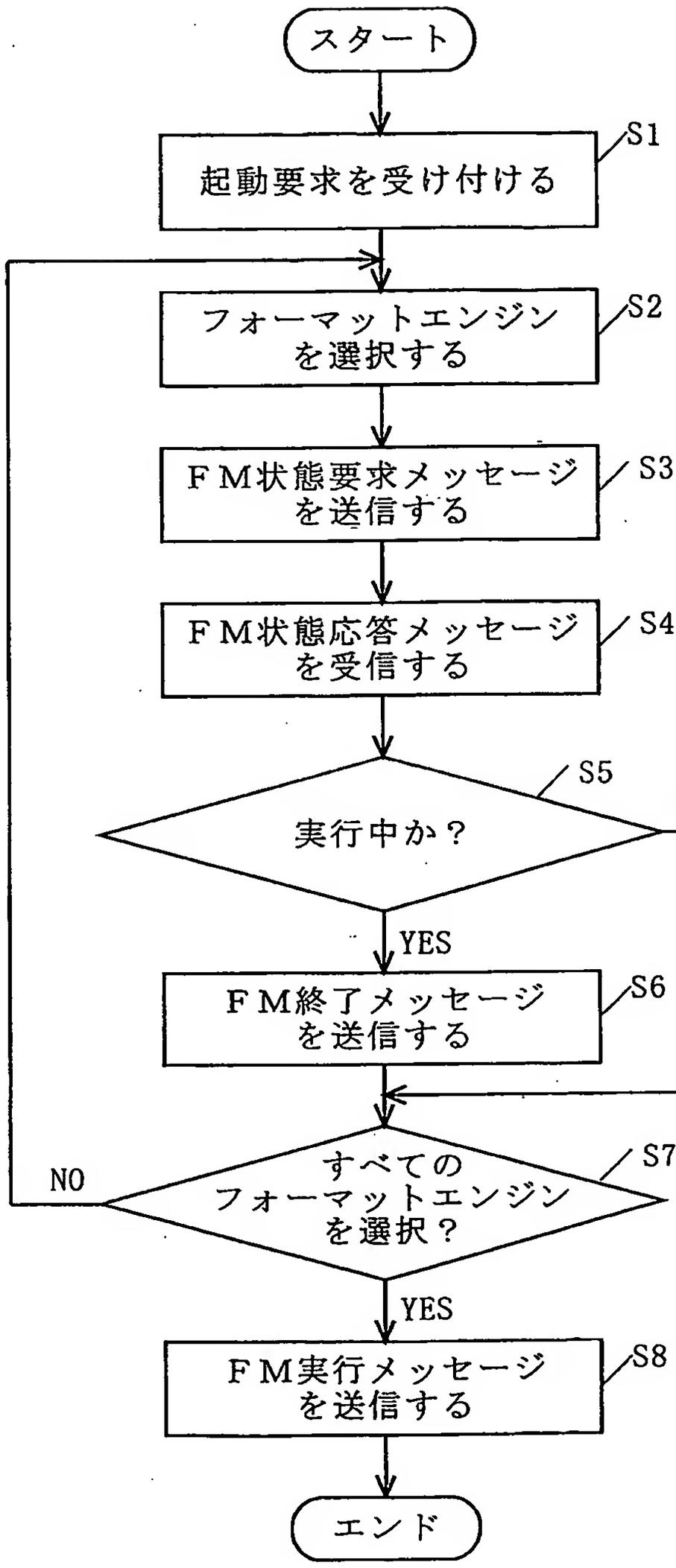

スタート
起動要求を受け付ける
S1
フォーマットエンジン
を選択する
S2
ＦＭ状態要求メッセージ
を送信する
S3
ＦＭ状態応答メッセージ
を受信する
S4
実行中か？
S5
NO
YES
ＦＭ終了メッセージ
を送信する
S6
すべての
フォーマットエンジン
を選択？
S7
NO
YES
ＦＭ実行メッセージ
を送信する
S8
エンド

図７５

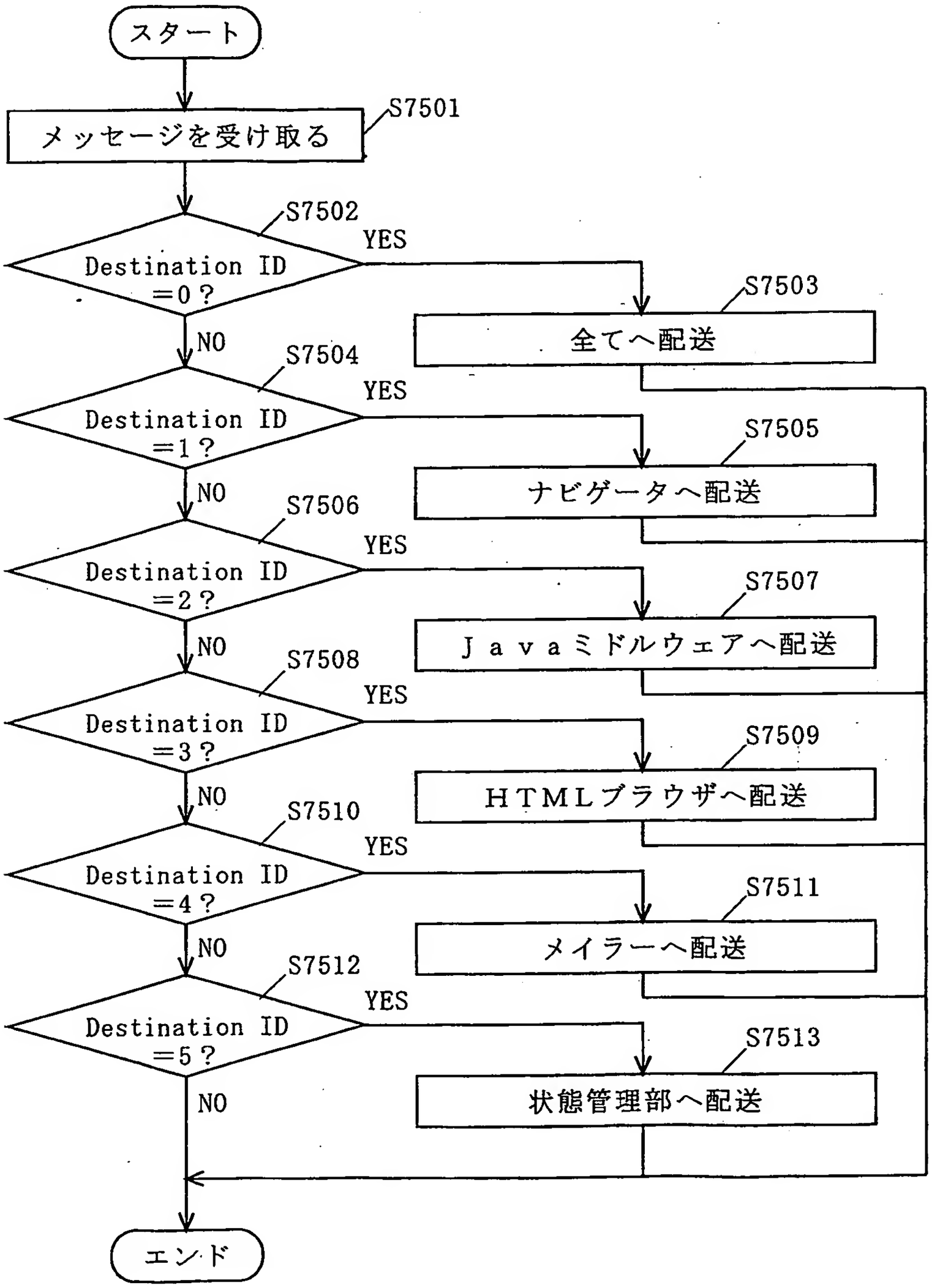

スタート
メッセージを受け取る
S7501
S7502
Destination ID =0？
YES
NO
全てへ配送
S7503
S7504
Destination ID =1？
YES
NO
ナビゲータへ配送
S7505
S7506
Destination ID =2？
YES
NO
Ｊａｖａミドルウェアへ配送
S7507
S7508
Destination ID =3？
YES
NO
ＨＴＭＬブラウザへ配送
S7509
S7510
Destination ID =4？
YES
NO
メイラーへ配送
S7511
S7512
Destination ID =5？
YES
NO
状態管理部へ配送
S7513
エンド

図７６

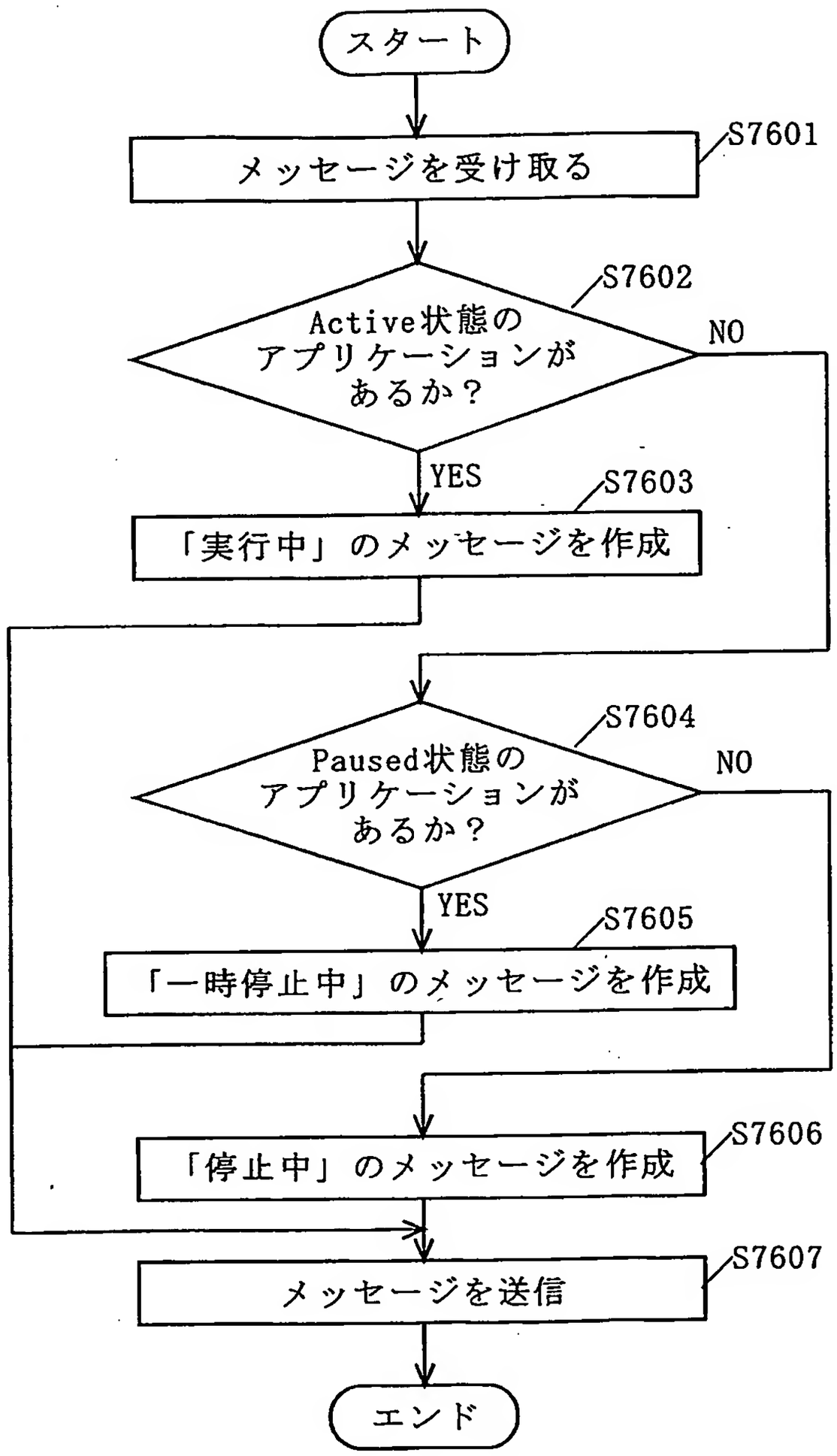

スタート
メッセージを受け取る
S7601
S7602
Active状態の
アプリケーションが
あるか？
NO
YES
S7603
「実行中」のメッセージを作成
S7604
Paused状態の
アプリケーションが
あるか？
NO
YES
S7605
「一時停止中」のメッセージを作成
「停止中」のメッセージを作成
S7606
メッセージを送信
S7607
エンド

図７７

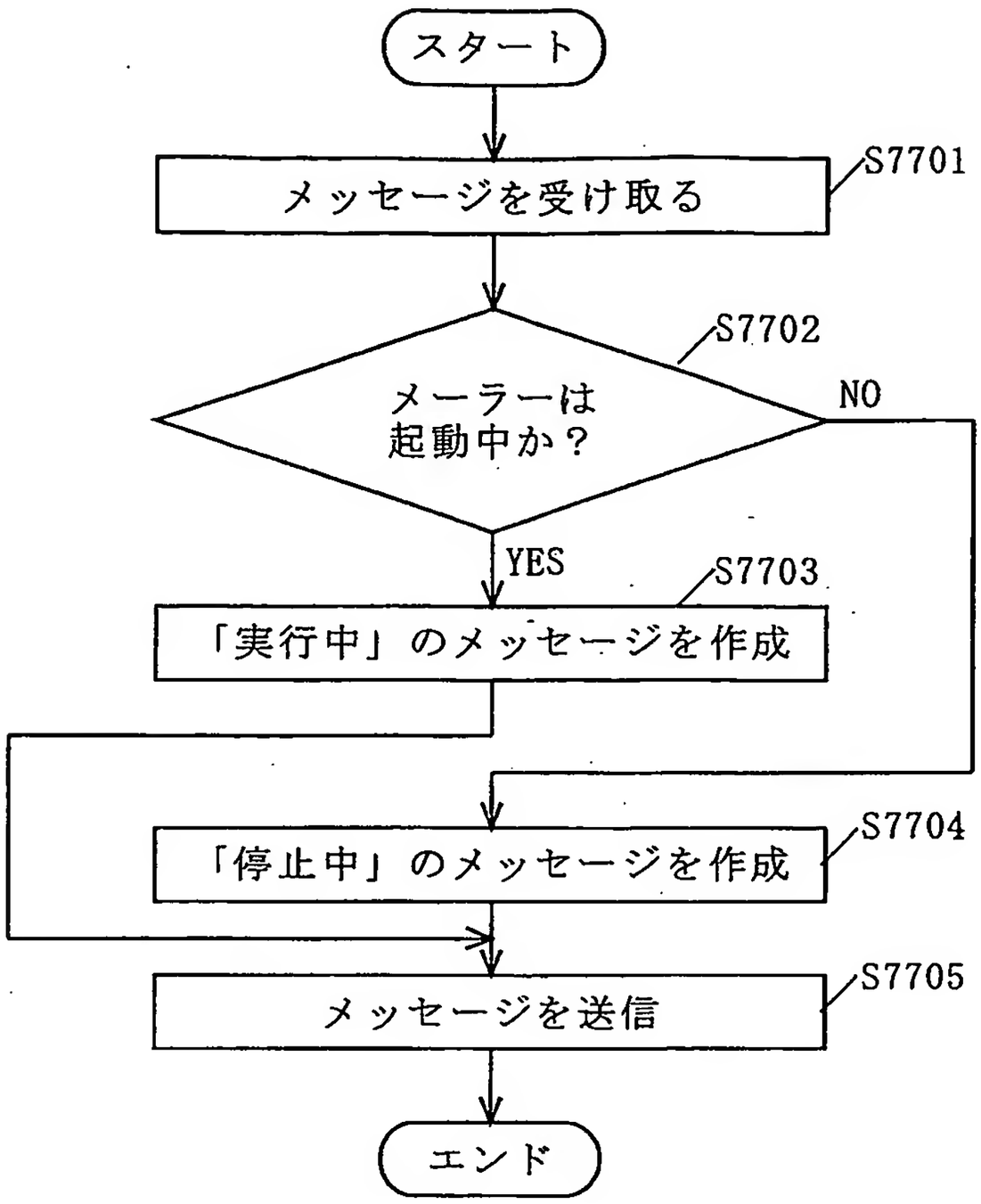

スタート
メッセージを受け取る
S7701
S7702
メーラーは
起動中か？
NO
YES
S7703
「実行中」のメッセージを作成
「停止中」のメッセージを作成
S7704
メッセージを送信
S7705
エンド

図７８

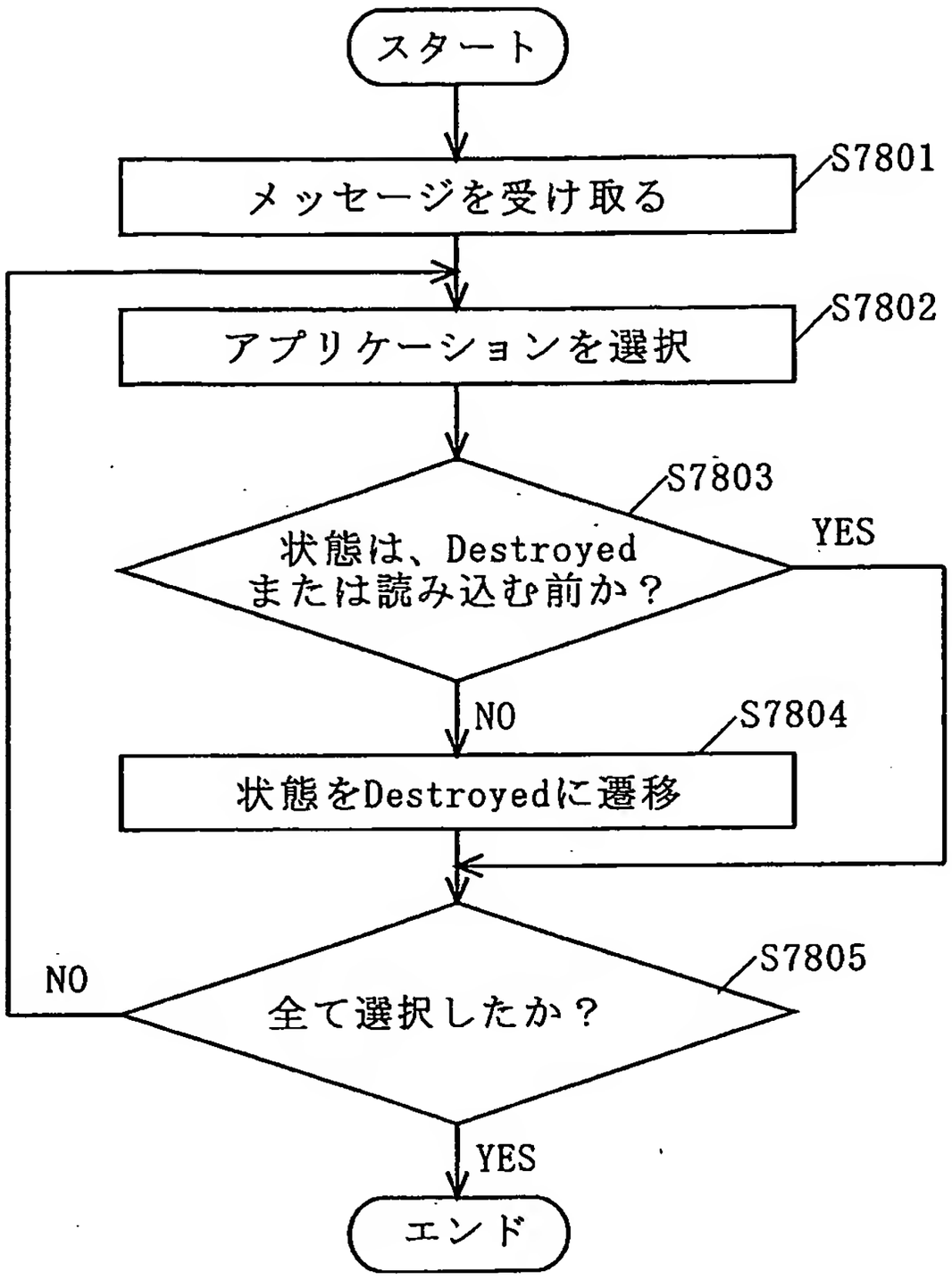

スタート
メッセージを受け取る
S7801
アプリケーションを選択
S7802
S7803
状態は、Destroyed
または読み込む前か？
YES
NO
S7804
状態をDestroyedに遷移
S7805
NO
全て選択したか？
YES
エンド

図７９

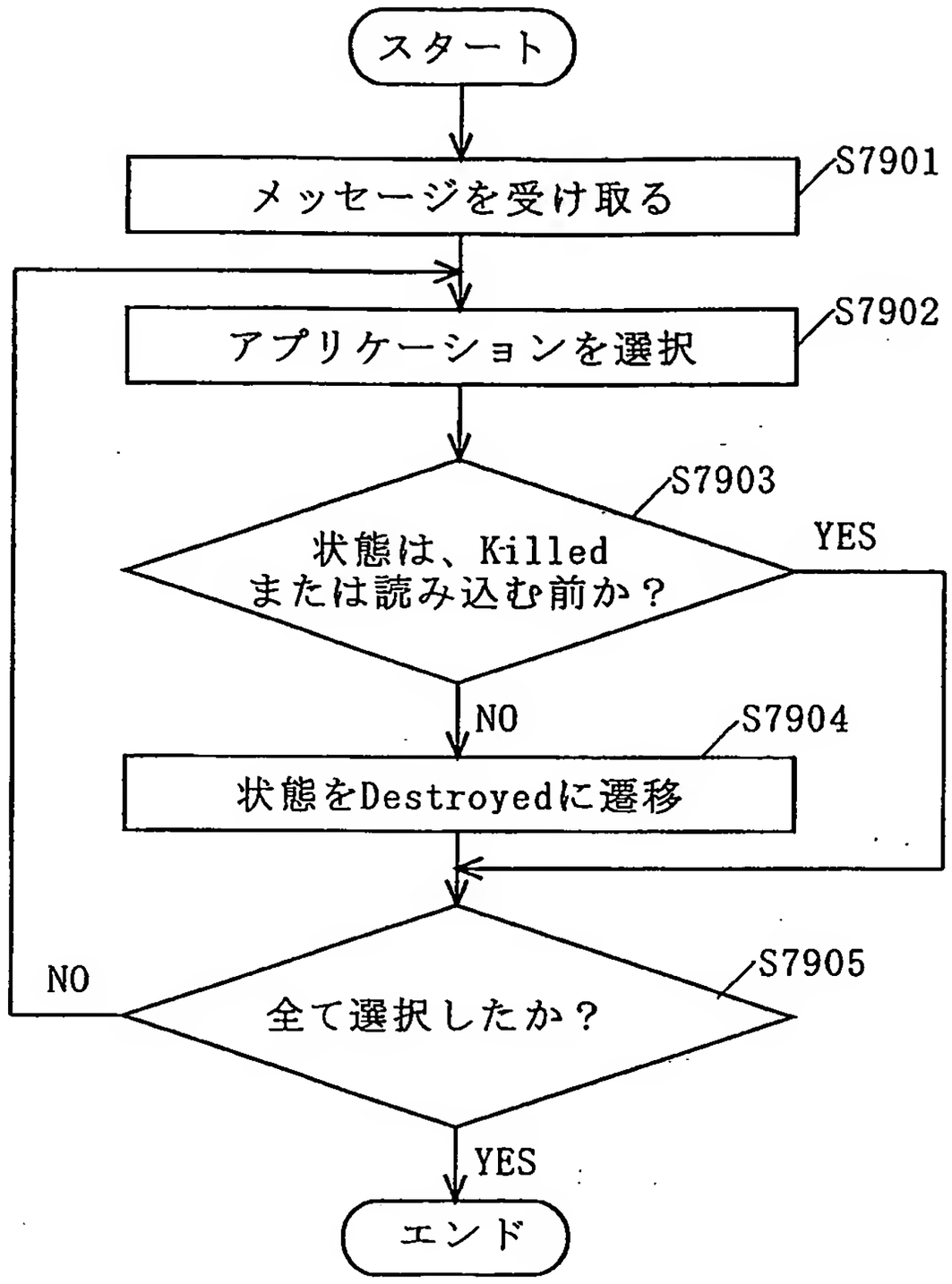

スタート
メッセージを受け取る
S7901
アプリケーションを選択
S7902
S7903
状態は、Killed
または読み込む前か？
YES
NO
S7904
状態をDestroyedに遷移
S7905
NO
全て選択したか？
YES
エンド

図８０

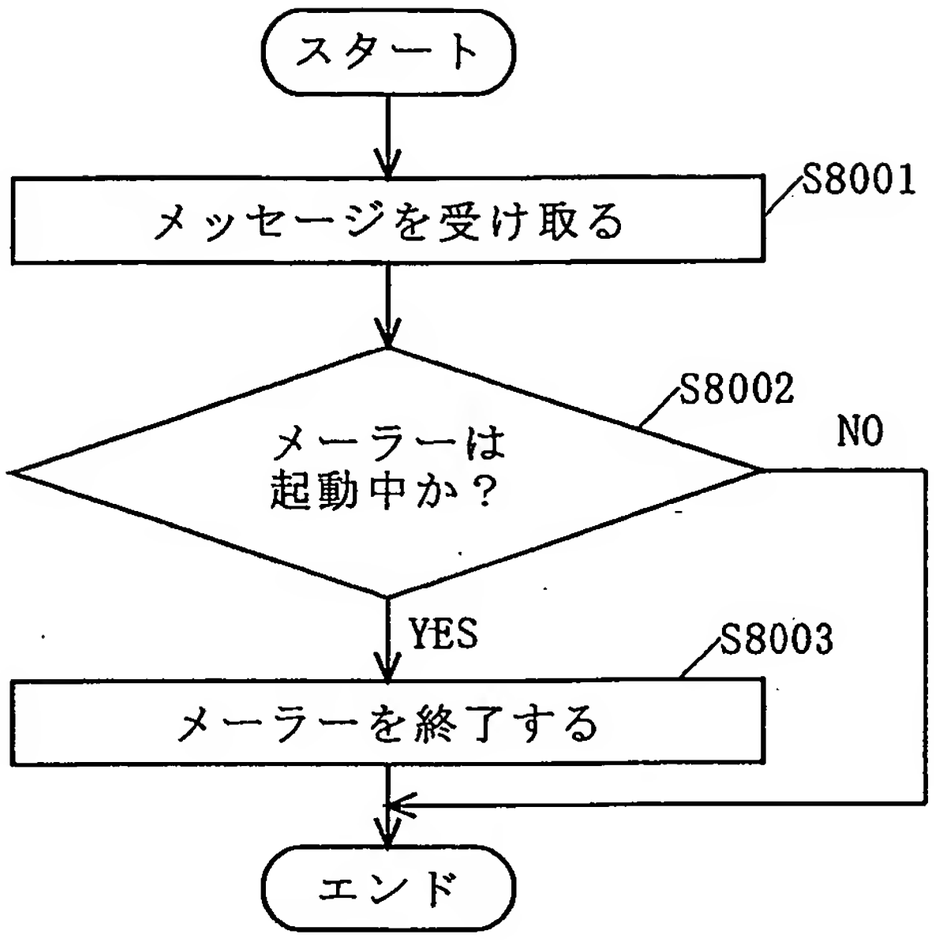

スタート
メッセージを受け取る
S8001
メーラーは
起動中か？
S8002
NO
YES
S8003
メーラーを終了する
エンド

図８１

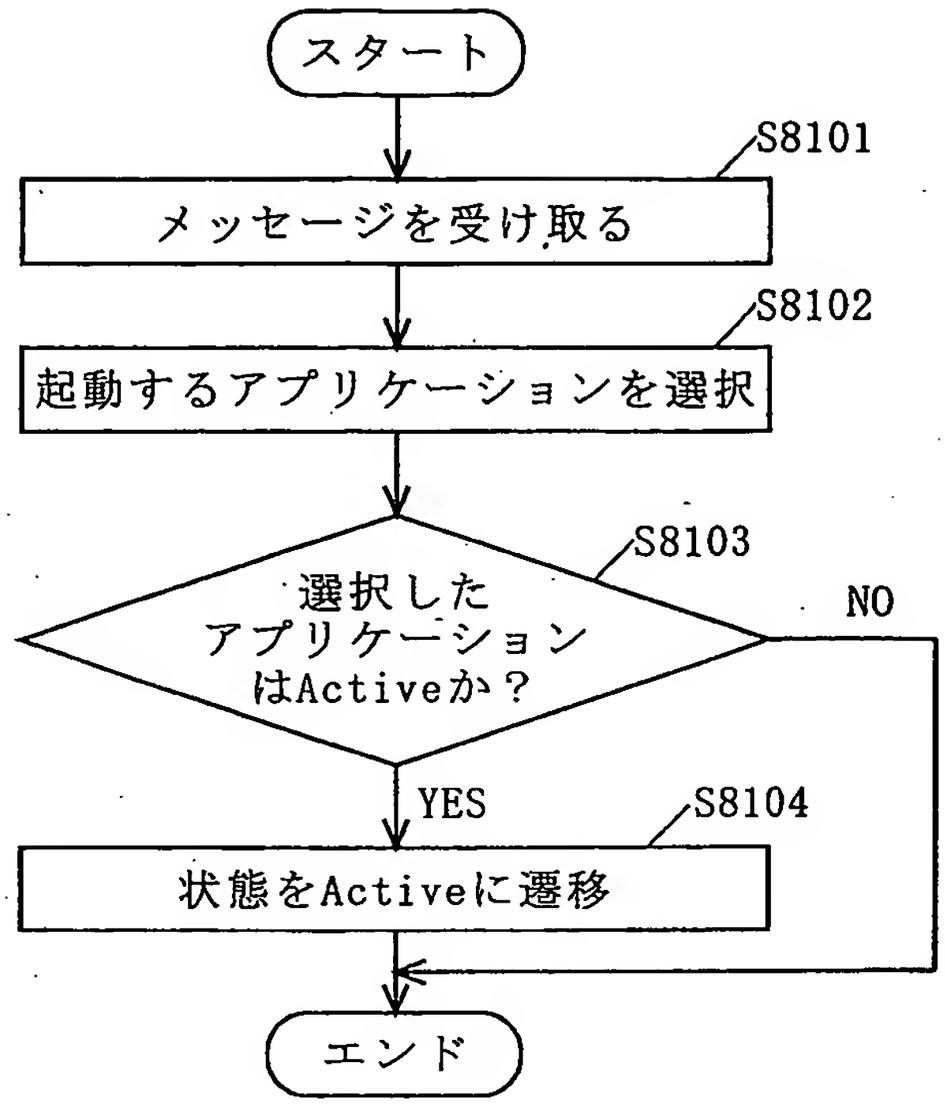

スタート
S8101
メッセージを受け取る
S8102
起動するアプリケーションを選択
S8103
選択した
アプリケーション
はActiveか？
NO
YES
S8104
状態をActiveに遷移
エンド

図８２

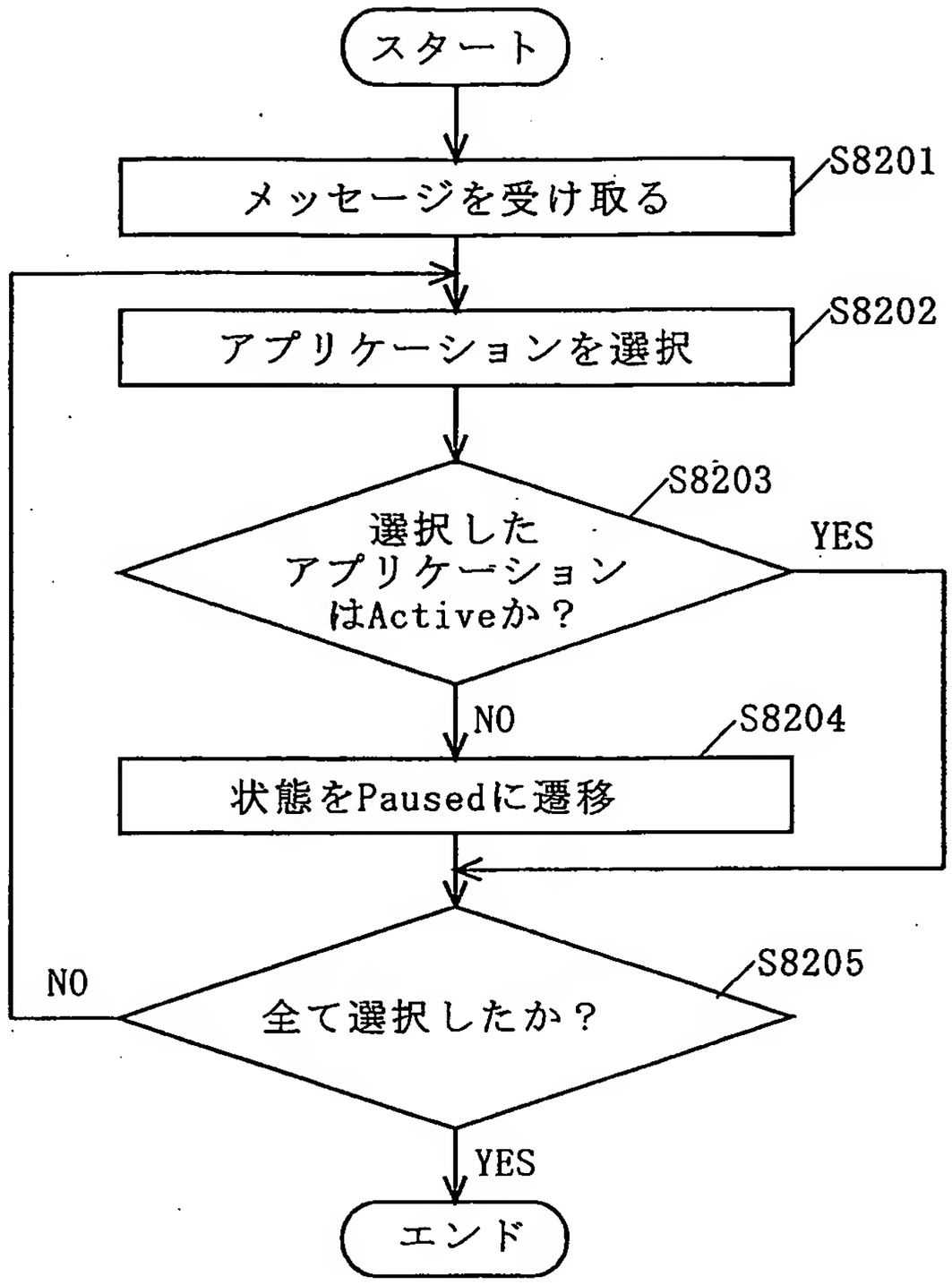

スタート
メッセージを受け取る
S8201
アプリケーションを選択
S8202
選択した
アプリケーション
はActiveか？
S8203
YES
NO
S8204
状態をPausedに遷移
全て選択したか？
S8205
NO
YES
エンド

図８３

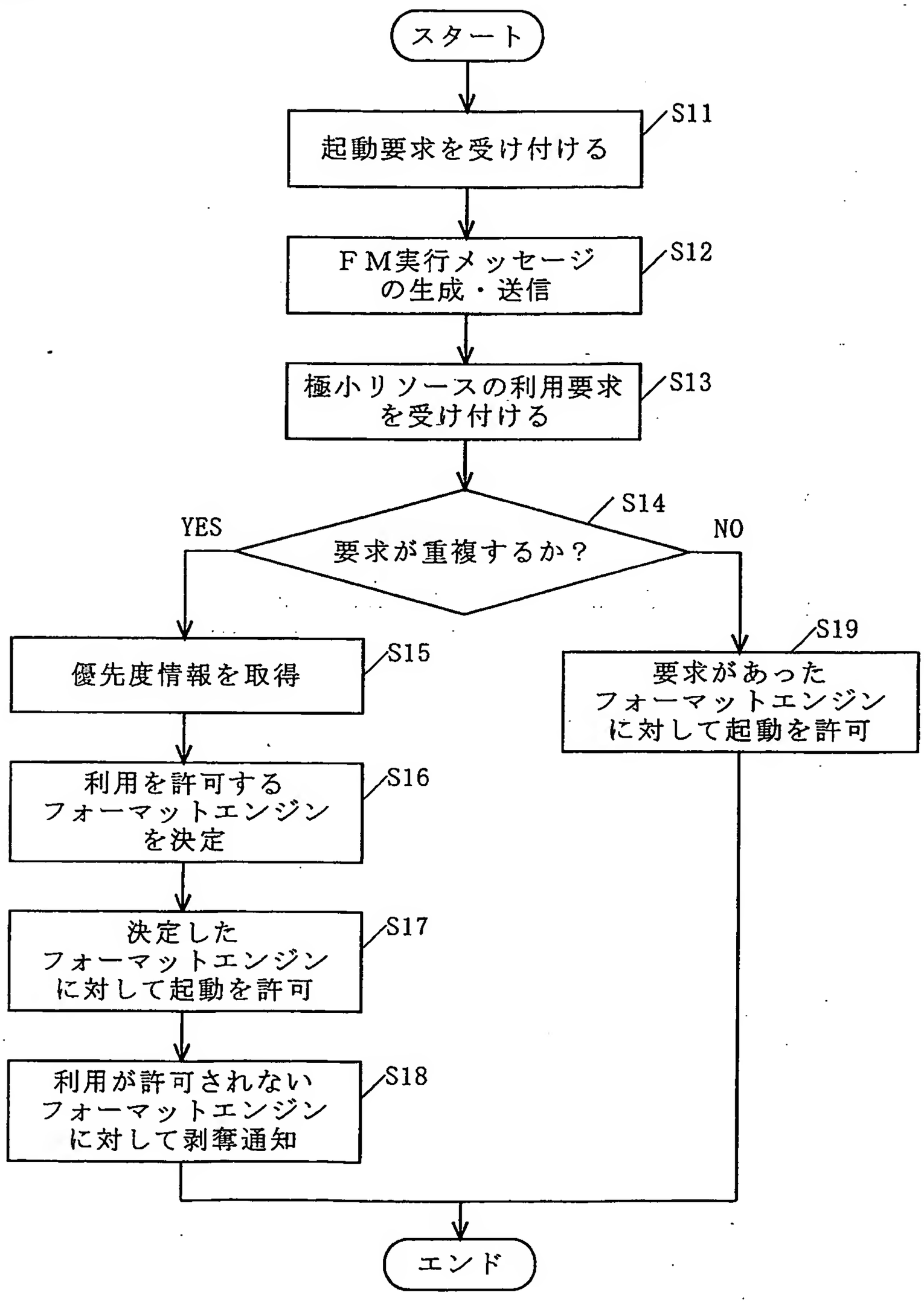

スタート
起動要求を受け付ける
S11
ＦＭ実行メッセージ
の生成・送信
S12
極小リソースの利用要求
を受け付ける
S13
要求が重複するか？
S14
YES
NO
優先度情報を取得
S15
利用を許可する
フォーマットエンジン
を決定
S16
決定した
フォーマットエンジン
に対して起動を許可
S17
利用が許可されない
フォーマットエンジン
に対して剥奪通知
S18
要求があった
フォーマットエンジン
に対して起動を許可
S19
エンド

図８４

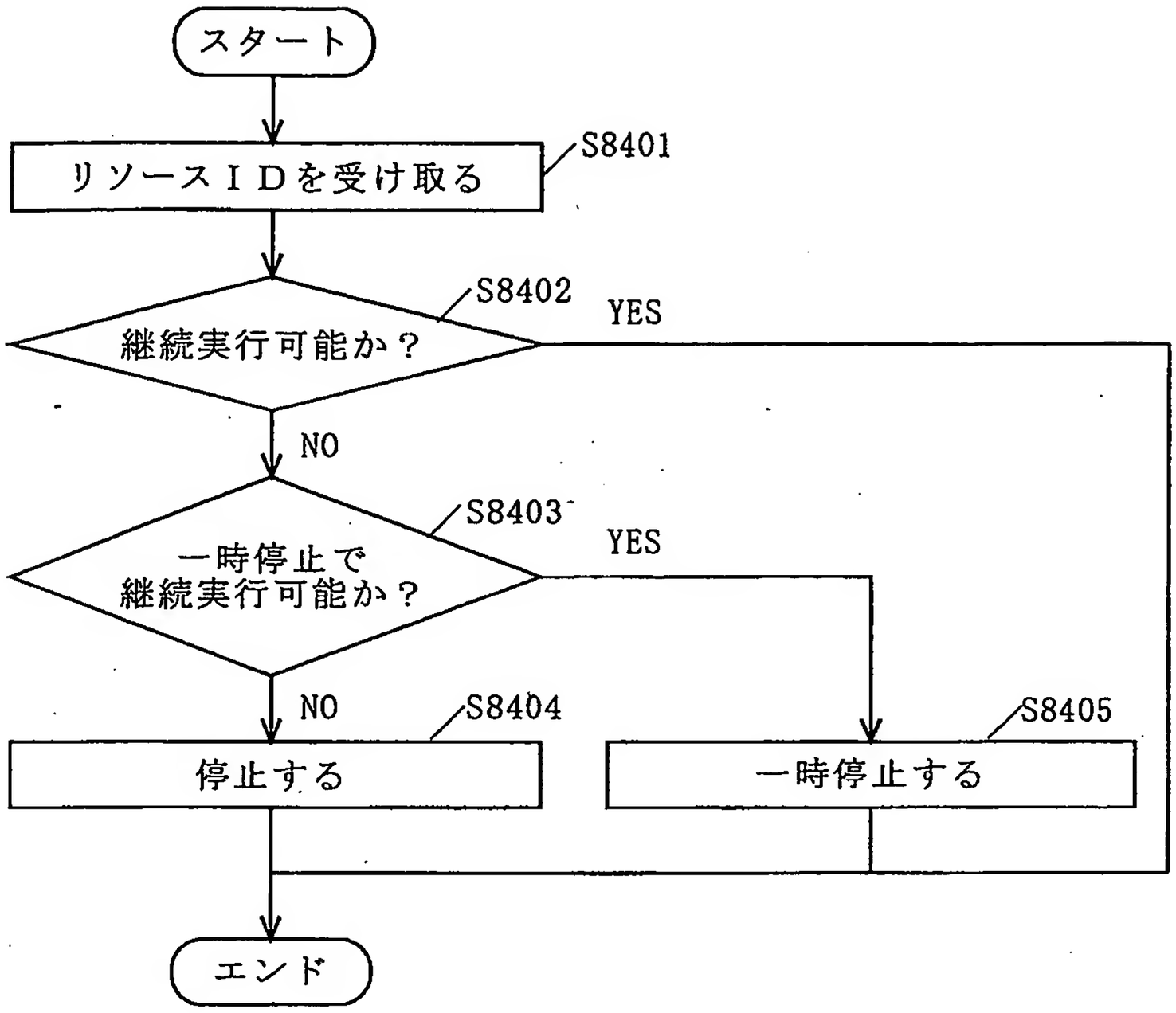

スタート
リソースＩＤを受け取る
S8401
継続実行可能か？
S8402
YES
NO
一時停止で
継続実行可能か？
S8403
YES
NO
S8404
S8405
停止する
一時停止する
エンド

図８５

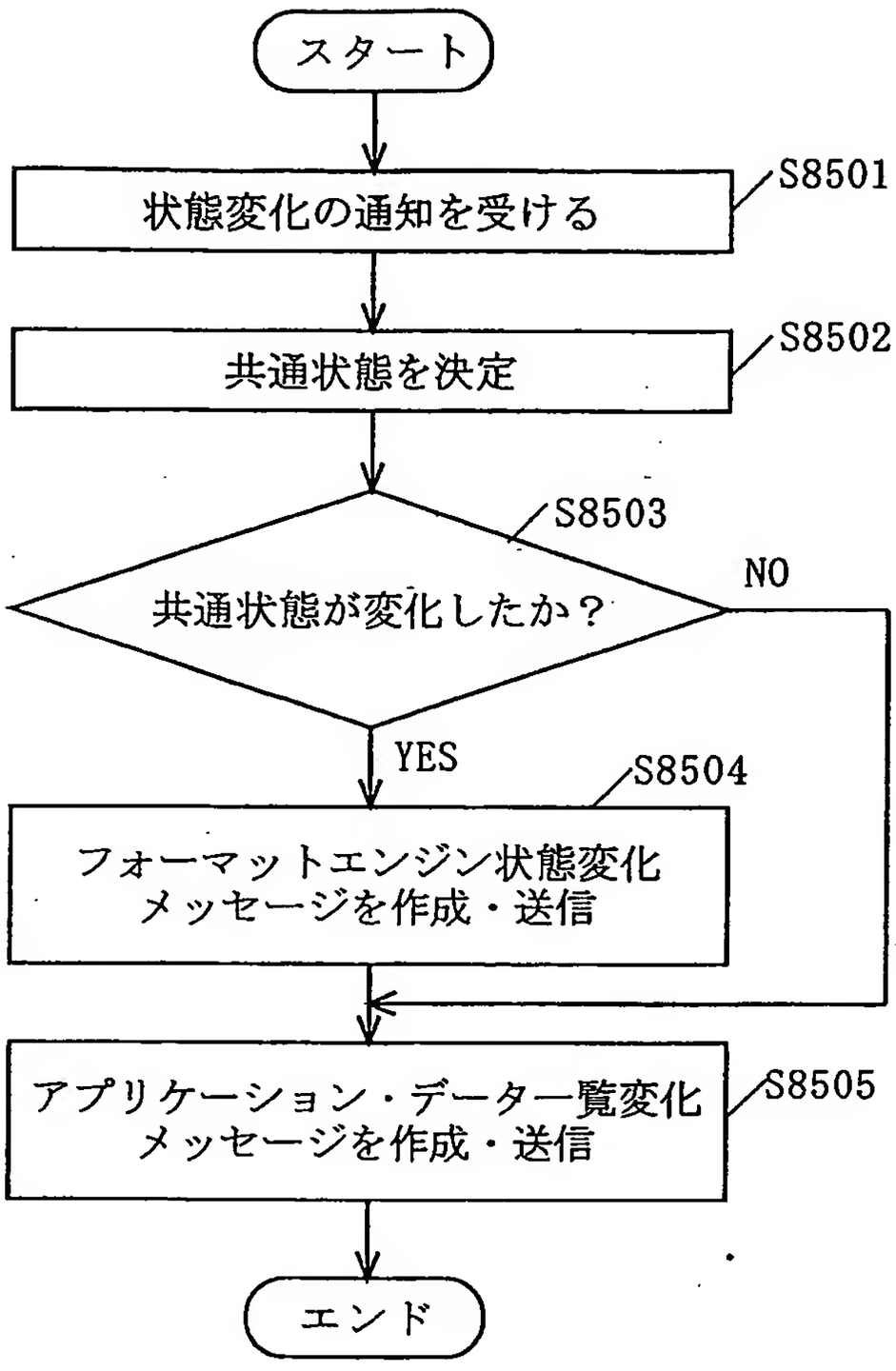

スタート
状態変化の通知を受ける
S8501
共通状態を決定
S8502
S8503
共通状態が変化したか？
NO
YES
S8504
フォーマットエンジン状態変化
メッセージを作成・送信
アプリケーション・データ一覧変化
メッセージを作成・送信
S8505
エンド

図８６

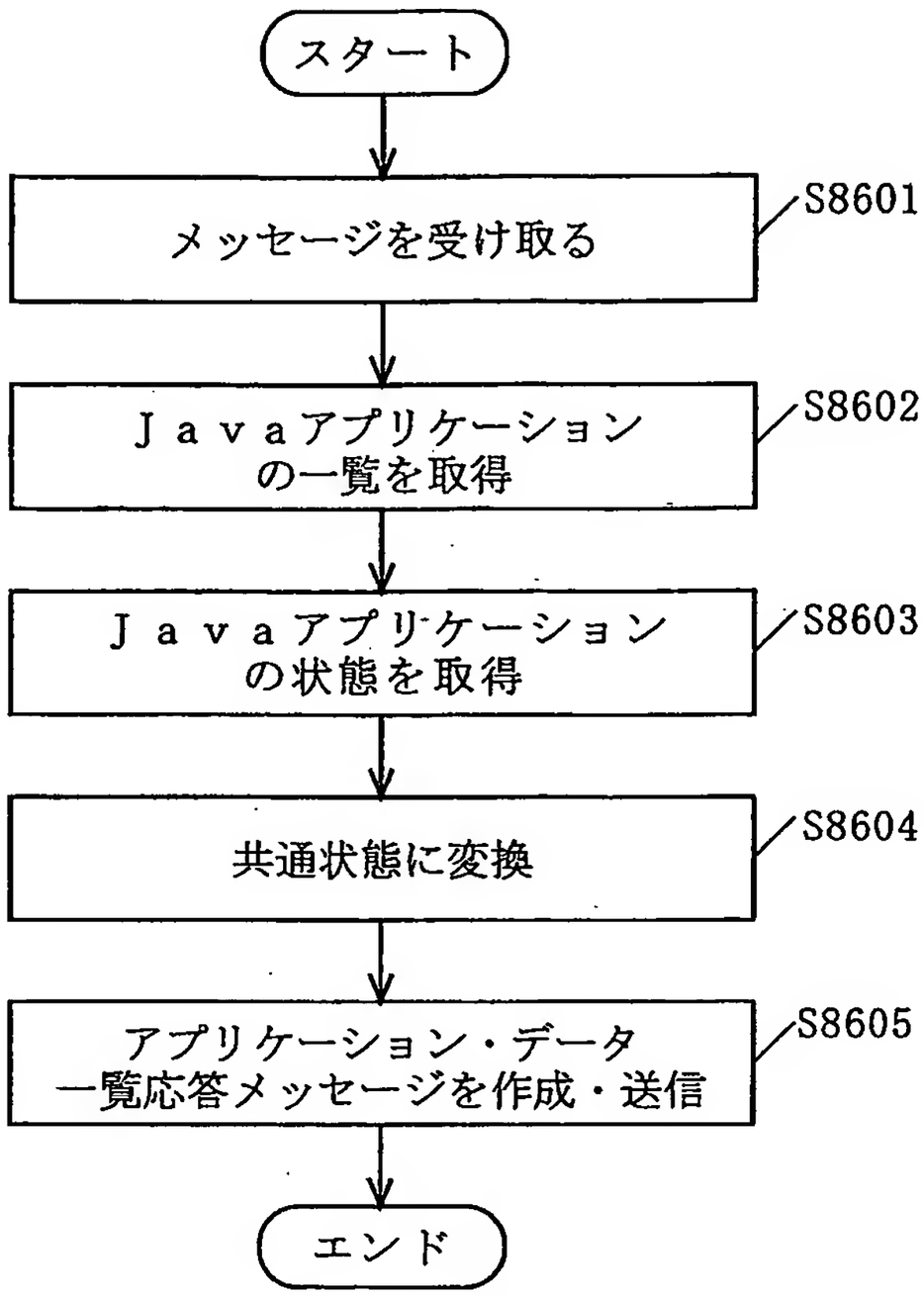

スタート
メッセージを受け取る
S8601
Ｊａｖａアプリケーション
の一覧を取得
S8602
Ｊａｖａアプリケーション
の状態を取得
S8603
共通状態に変換
S8604
アプリケーション・データ
一覧応答メッセージを作成・送信
S8605
エンド

図８７

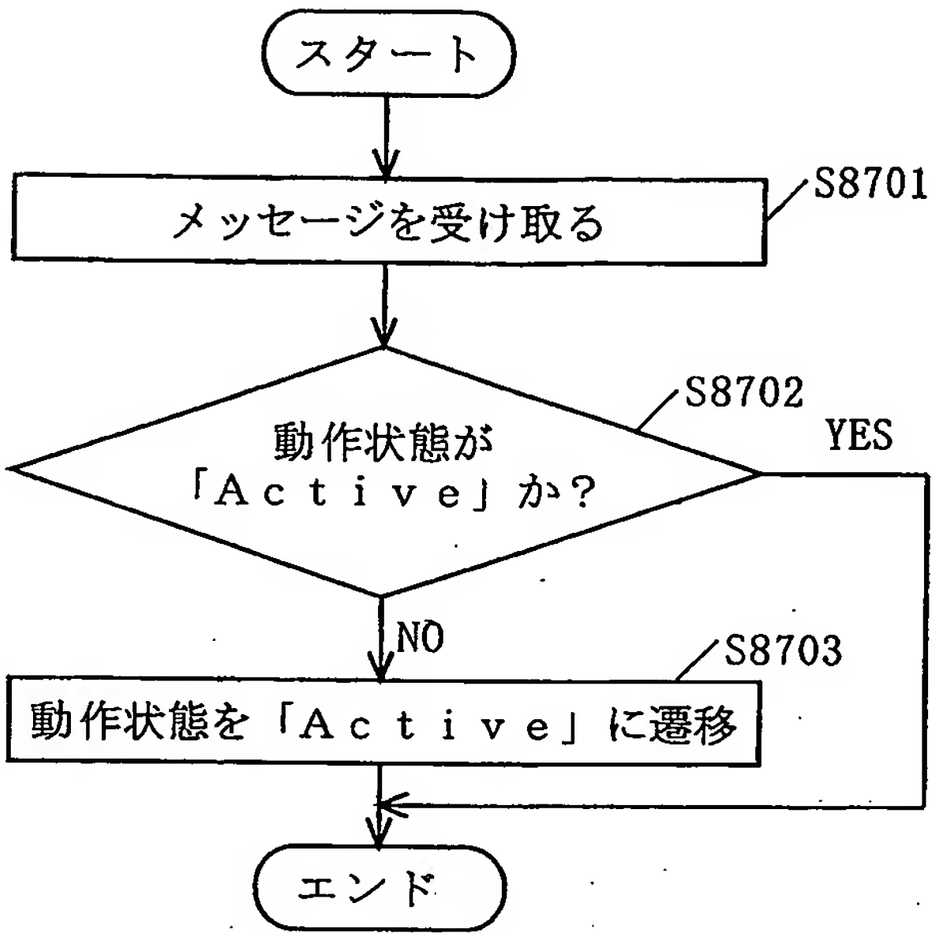

スタート
メッセージを受け取る
S8701
動作状態が
「Ａｃｔｉｖｅ」か？
S8702
YES
NO
S8703
動作状態を「Ａｃｔｉｖｅ」に遷移
エンド

図８８

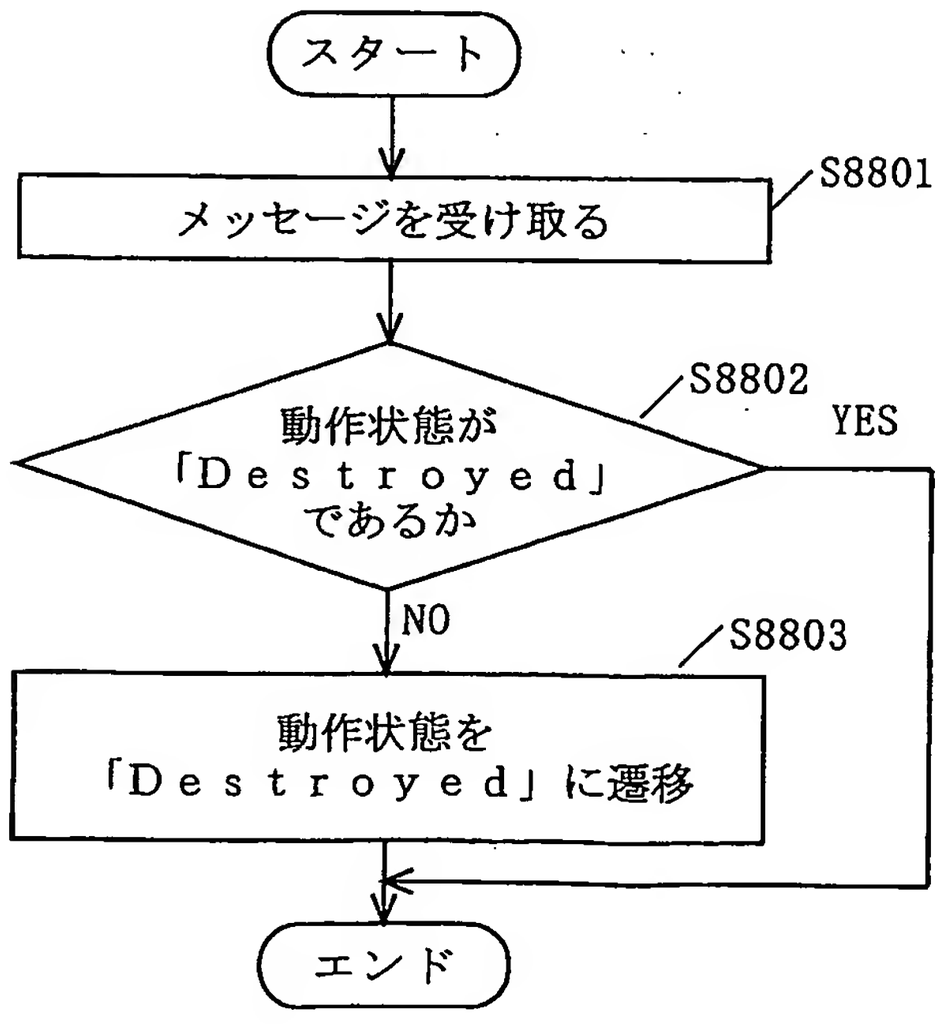

スタート
メッセージを受け取る
S8801
動作状態が
「Ｄｅｓｔｒｏｙｅｄ」
であるか
S8802
YES
NO
S8803
動作状態を
「Ｄｅｓｔｒｏｙｅｄ」に遷移
エンド

図８９

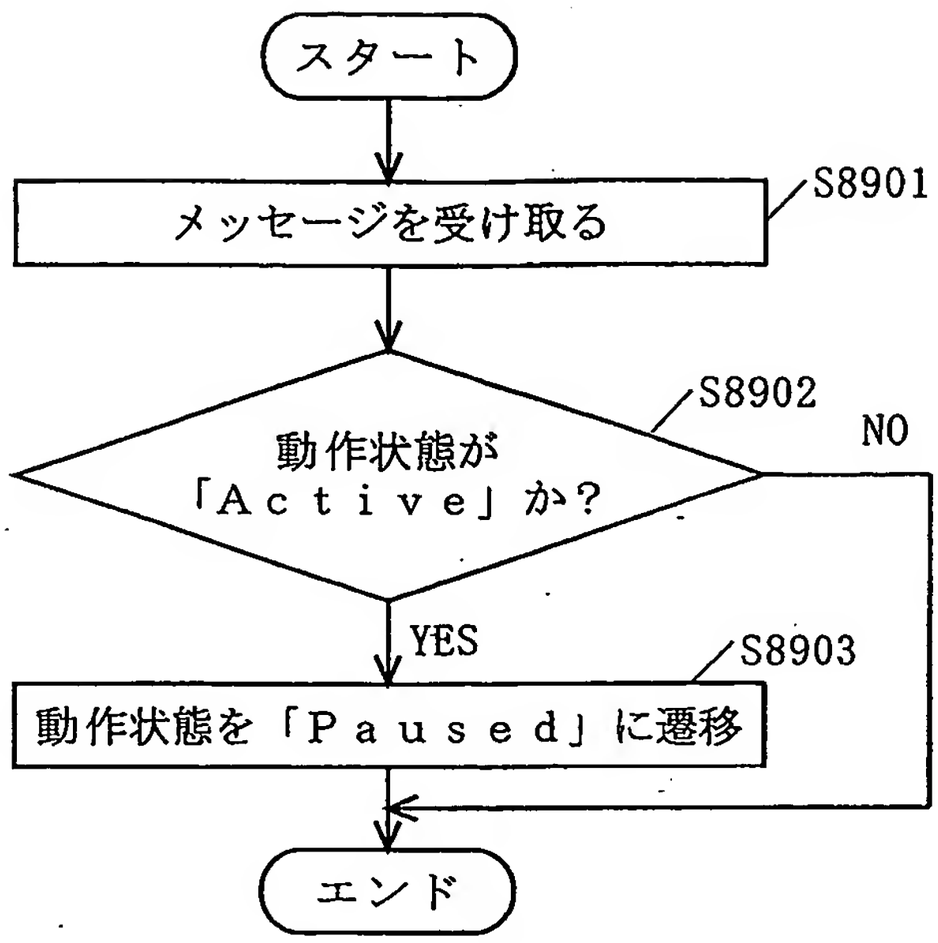

スタート
メッセージを受け取る
S8901
動作状態が
「Ａｃｔｉｖｅ」か？
S8902
NO
YES
S8903
動作状態を「Ｐａｕｓｅｄ」に遷移
エンド

図９０

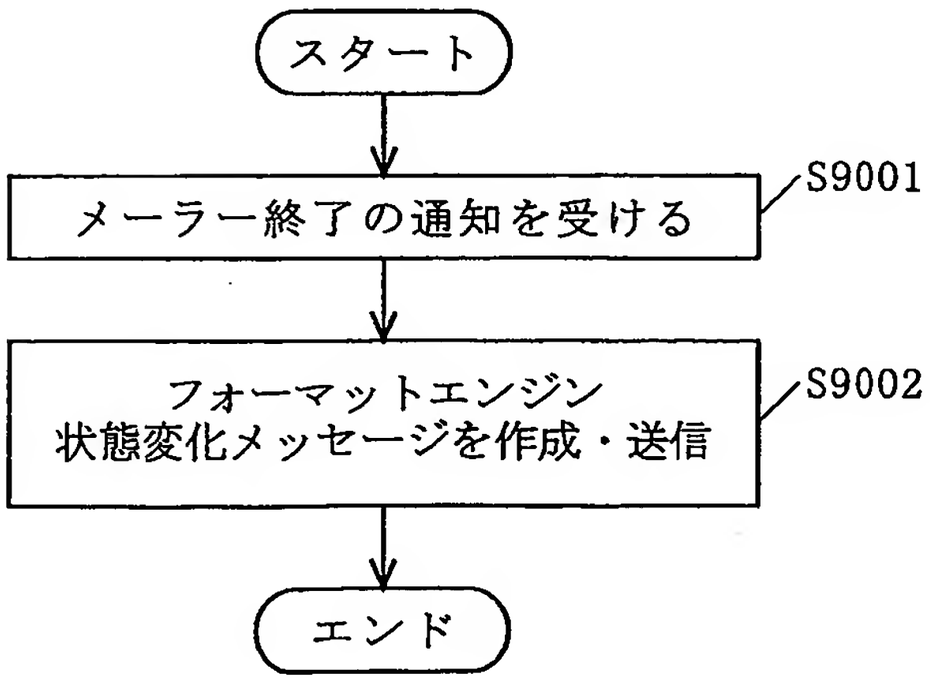

スタート
メーラー終了の通知を受ける
S9001
フォーマットエンジン
状態変化メッセージを作成・送信
S9002
エンド

図９１

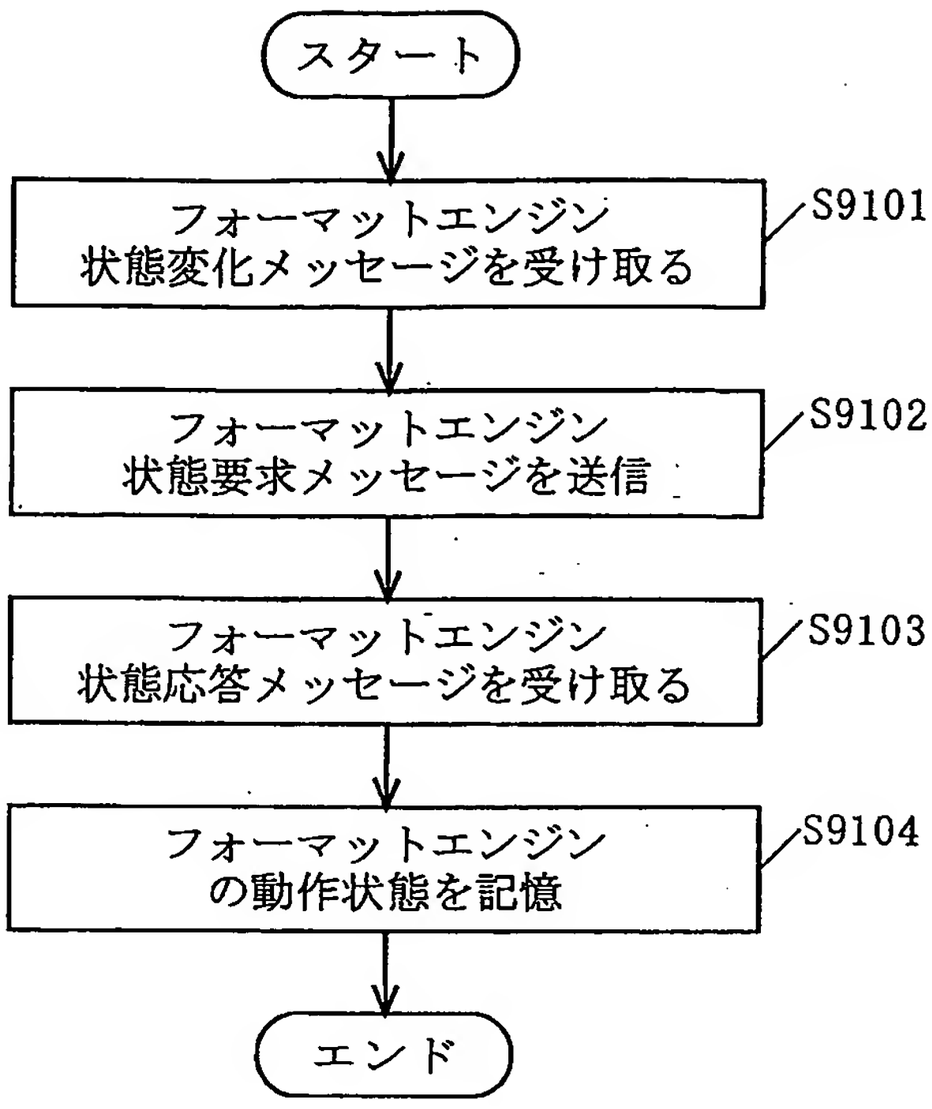

スタート
フォーマットエンジン
状態変化メッセージを受け取る
S9101
フォーマットエンジン
状態要求メッセージを送信
S9102
フォーマットエンジン
状態応答メッセージを受け取る
S9103
フォーマットエンジン
の動作状態を記憶
S9104
エンド

図９２

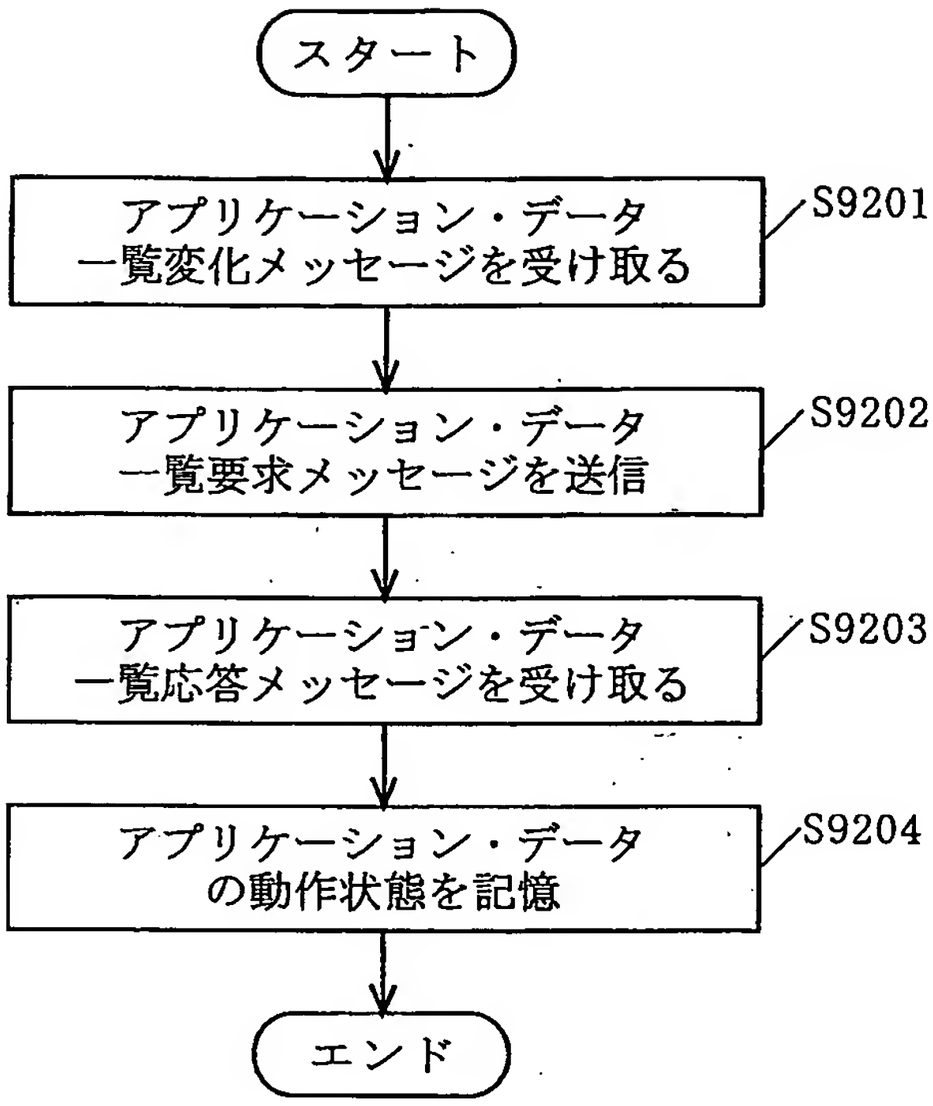

スタート
アプリケーション・データ
一覧変化メッセージを受け取る
S9201
アプリケーション・データ
一覧要求メッセージを送信
S9202
アプリケーション・データ
一覧応答メッセージを受け取る
S9203
アプリケーション・データ
の動作状態を記憶
S9204
エンド

図９３

Ｊａｖａミドルウェア	ＨＴＭＬブラウザ	メーラー
停止中	停止中	停止中
一時停止中	停止中	停止中
停止中	一時停止中	停止中
一時停止中	一時停止中	停止中

図９４

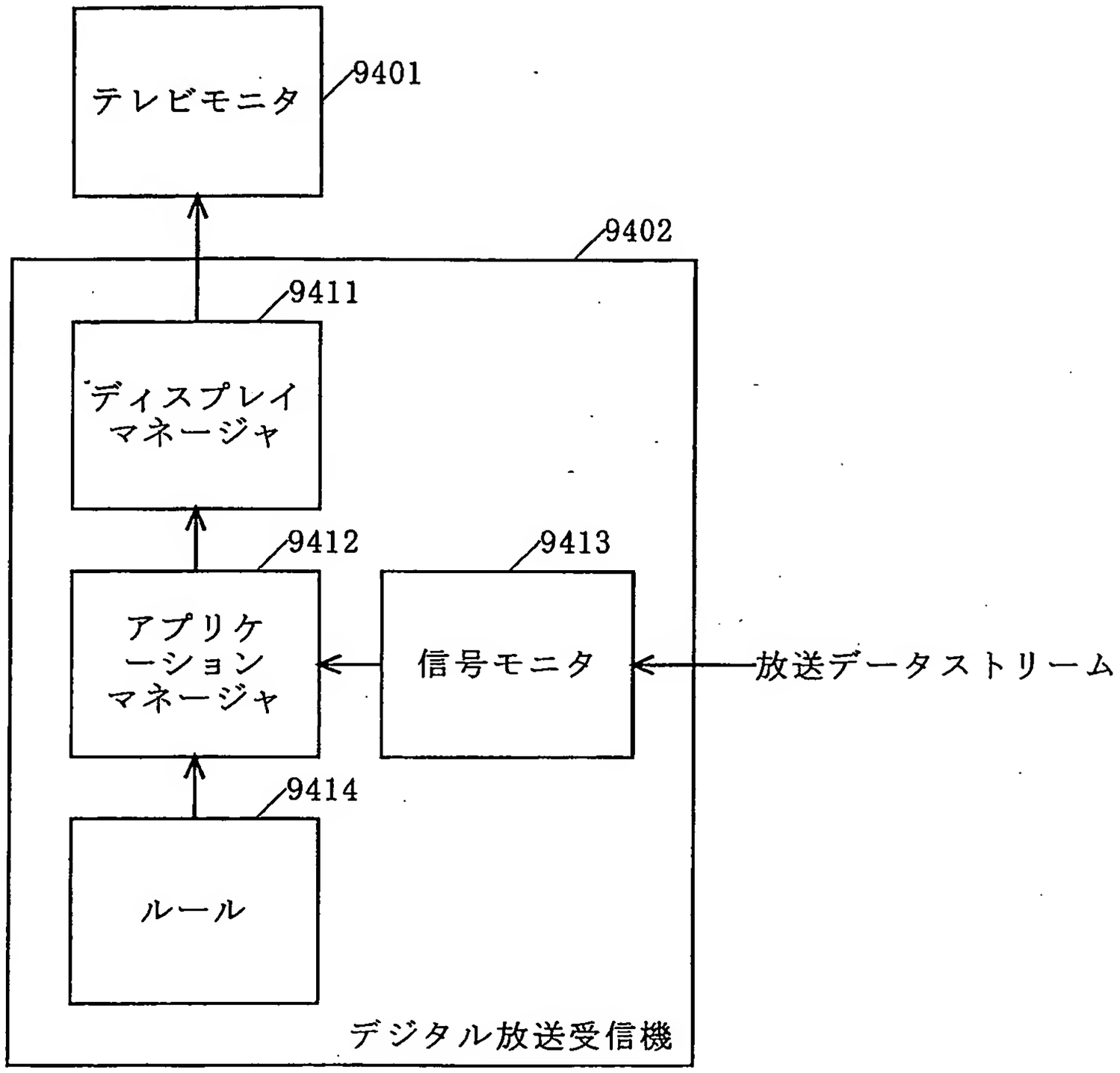
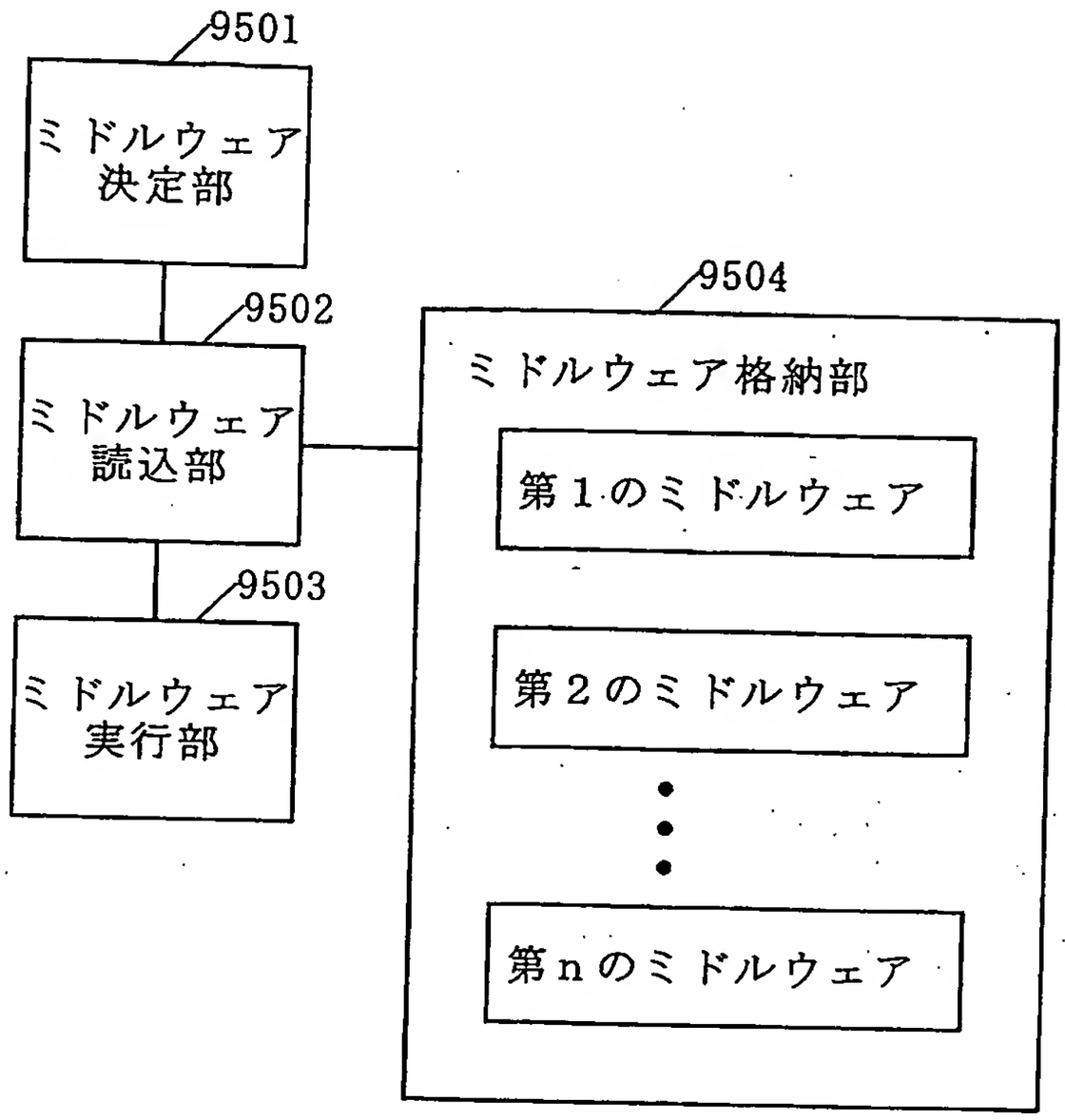

テレビモニタ
9401
9402
9411
ディスプレイ
マネージャ
9412
9413
アプリケ
ーション
マネージャ
信号モニタ
放送データストリーム
9414
ルール
デジタル放送受信機

図９５


9501
ミドルウェア決定部
9502
ミドルウェア読込部
9503
ミドルウェア実行部
9504
ミドルウェア格納部
第１のミドルウェア
第２のミドルウェア
第ｎのミドルウェア

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP03/12932

A. CLASSIFICATION OF SUBJECT MATTER
Int.Cl[7] H04N5/44, 5/93

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
Int.Cl[7] H04N5/38-5/46, 5/91-5/956, 5/78-5/784, G06F3/00, 13/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Jitsuyo Shinan Koho	1922-1996	Toroku Jitsuyo Shinan Koho	1994-2003
Kokai Jitsuyo Shinan Koho	1971-2003	Jitsuyo Shinan Toroku Koho	1996-2003

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y A	JP 7-44477 A (Canon Inc.), 14 February, 1995 (14.02.95), Full text & EP 637157 A & US 5889193 A	1-3,8 4-7
Y A	JP 11-317991 A (Toshiba Corp.), 16 November, 1999 (16.11.99), Full text & EP 1020858 A & WO 99/18574 A1	1-3,8 4-7
A	JP 2002-238003 A (Matsushita Electric Industrial Co., Ltd.), 23 August, 2002 (23.08.02), Full text (Family: none)	1-8

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search 22 December, 2003 (22.12.03)	Date of mailing of the international search report 20 January, 2004 (20.01.04)
Name and mailing address of the ISA/ Japanese Patent Office Facsimile No.	Authorized officer Telephone No.

Form PCT/ISA/210 (second sheet) (July 1998)

国際調査報告

国際出願番号　ＰＣＴ／ＪＰ０３／１２９３２

A.　発明の属する分野の分類（国際特許分類（ＩＰＣ））
Ｉｎｔ.ｃｌ'Ｈ０４Ｎ５／４４，５／９３

B.　調査を行った分野
調査を行った最小限資料（国際特許分類（ＩＰＣ））
Ｉｎｔ.ｃｌ'Ｈ０４Ｎ５／３８－５／４６，５／９１－５／９５６，５／７８－５／７８４，
Ｇ０６Ｆ３／００，１３／００

最小限資料以外の資料で調査を行った分野に含まれるもの
日本国実用新案公報　　１９２２－１９９６年
日本国公開実用新案公報１９７１－２００３年
日本国登録実用新案公報１９９４－２００３年
日本国実用新案登録公報１９９６－２００３年

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

C.　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求の範囲の番号
Y A	ＪＰ ７－４４４７７ Ａ(キャノン株式会社) １９９５.０２.１４，全文 & ＥＰ ６３７１５７ Ａ & ＵＳ ５８８９１９３ Ａ	1-3，8 4-7
Y A	ＪＰ １１－３１７９９１ Ａ(株式会社東芝) １９９９.１１.１６，全文 & ＥＰ １０２０８５８ Ａ & ＷＯ ９９／１８５７４ Ａ１	1-3，8 4-7
A	ＪＰ ２００２－２３８００３ Ａ(松下電器産業株式会社) ２００２.０８.２３，全文(ファミリーなし)	1-8

☐ C欄の続きにも文献が列挙されている。　　☐ パテントファミリーに関する別紙を参照。

＊　引用文献のカテゴリー
「A」特に関連のある文献ではなく、一般的技術水準を示すもの
「E」国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの
「L」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）
「O」口頭による開示、使用、展示等に言及する文献
「P」国際出願日前で、かつ優先権の主張の基礎となる出願の日の後に公表された文献
「T」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの
「X」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
「Y」特に関連のある文献であって、当該文献と他の１以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
「&」同一パテントファミリー文献

国際調査を完了した日	国際調査報告の発送日
２２．１２．０３	20.1.2004
国際調査機関の名称及びあて先 日本国特許庁（ＩＳＡ／ＪＰ） 郵便番号１００－８９１５ 東京都千代田区霞が関三丁目４番３号	特許庁審査官（権限のある職員） 西谷 憲人　印　　５Ｐ　９１８７ 電話番号　０３－３５８１－１１０１　内線　３５８１

様式ＰＣＴ／ＩＳＡ／２１０（第２ページ）（１９９８年７月）